ioProgrammo

TECH-ED 2004 REPORTAGE ESCLUSIVO | **ECLIPSE 3.0** AL VIA IL CORSO A PUNTATE

VERSIONE PLUS ✔ RIVISTA+LIBRO+CD €14,90 | VERSIONE STANDARD ☐ RIVISTA+CD €6,90

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • OTTOBRE 2004 • ANNO VIII, N.9 (84)

SPECIALE

NUOVA LINFA ALLE TUE APPLICAZIONI!

Programmazione a oggetti: cosa cambierà

- ✔ La nuova sintassi
- ✔ Gli esempi di codice da utilizzare subito
- ✔ Le tecniche di sviluppo più avanzate

IN ESCLUSIVA

Database in Visual Basic

Report professionali: impariamo a realizzarli e stamparli nelle nostre applicazioni

INTERNET COME FONTE DI DATI

Importare le quotazioni di borsa dalle pagine Web

HACKING PER PRINCIPIANTI

Metasploit: un Wizard per realizzare exploit

INTERNET
- Sviluppare per i-Mode

ELETTRONICA
- Etichette RFID: i dettagli di un successo
- Un voltimetro digitale nel PC

SMARTPHONE
- Leggere i dati della SIM

CORSI
- Java: le interfacce
- VB.NET: file e directory
- C#: migliorare il debug

SERVIZI WEB
- Pubblicare documenti Word su Internet tramite RSS

ADVANCED
- Serializzare le classi in .NET
- Apache Velocity: il codice in automatico

SOLUZIONI
- Scacchiere: strategie e simulazioni
- Giochiamo con i numeri

ISSN 1128-594X
9 771128 594641 40084
ioProgrammo Anno VIII - N° 9 (84) • € 14,90
EDIZIONI MASTER
www.edmaster.it

C-ROBOTS 2004: IL REGOLAMENTO PER PARTECIPARE ALLA NUOVA SFIDA

CONTENTS ▼ **Questo mese su ioProgrammo**

Anno VIII - n. 9 (84) Ottobre 2004

▼ Un forte aroma di Java

Questo mese trovate numerosi articoli dedicati a Java. Vorrei richiamare la vostra attenzione specialmente sull'articolo di copertina e sulla prima puntata del corso su Eclipse.
Java 1.5 è alle porte e, mentre scrivo, dal sito della Sun è possibile effettuare il download della RC. Purtroppo, le policy della Sun impediscono la distribuzione del JDK prima che sia rilasciato ufficialmente, questo il motivo per cui nel CD allegato trovate unicamente la versione 1.4. Le innovazioni della nuova versione sono numerose e tutte di forte impatto. Certo, alla Sun hanno dovuto fare una difficile quadratura del cerchio: arricchire un linguaggio che deve il suo successo alla sua rivoluzionaria semplicità non deve essere stato semplice. Il lavoro che hanno fatto direi che è eccellente, anche se, per dirla tutta, credo che il codice paghi qualcosa in leggibilità. Inutile dire che il nostro sforzo sarà quello di accompagnarvi nel miglior utilizzo delle innovazioni introdotte e l'ottimo articolo di Joe Naccarato ne è la testimonianza. Ne seguiranno altri. Abbiamo tentato di riassumere il più possibile le innovazioni per mettervi in grado subito di cominciare con il piede giusto ma, per una evoluzione così importante non si può riassumere in un singolo articolo.
A testimoniare l'importanza del rilascio c'è il cambio di versione introdotto da Sun: in futuro, non ci si dovrà più riferire a Java 1.5, bensì a Java 5. Certo, c'è molto di marketing in questa scelta ma, per una volta, il mercato aiuta a fare chiarezza.

Raffaele del Monaco

All'inizio di ogni articolo, troverete un nuovo simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.
Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **hana-bi** *Password:* **sonatine**

ioProgrammo

Anno VIII - N.ro 9 (84) - Ottobre 2004 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale Massimo Sesti
Direttore Responsabile Massimo Sesti
Responsabile Editoriale Gianmarco Bruni
Responsabile Marketing Antonio Meduri
Editor Gianfranco Forlino
Coordinamento redazionale Raffaele del Monaco
Redazione Domenico Pingitore, Thomas Zaffino
Collaboratori S. Ascheri, M. Autiero, L. Buono, M. Camangi, M. Casario, E. Florio, F. Grimaldi, A. Lippo, F. Lippo, M. Locuratolo, P. Martemucci, M. Messina, F. Mestrone, G. Naccarato, C. Pelliccia, P. Perrotta, M. Poponi, L. Spuntoni, E. Tavolaro, F. Vaccaro, V. Vessia, D. Visicchio.
Segreteria di Redazione Veronica Longo

Realizzazione grafica Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri, R. Del Bo
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale SET S.r.l.
Coordinamento Tecnico Piero Mannelli
Realizzazione CD-Rom Paolo Iacona
Pubblicità Master Advertising s.r.l.
Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite Daisy Zonato

Editore Edizioni Master S.r.l.
Sede di Milano: Via Cesare Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Amministratore Unico: Massimo Sesti

Abbonamento e arretrati
ITALIA: Abbonamento Annuale: ioProgrammo Basic (11 numeri) €52,90 sconto 30% sul prezzo di copertina di €75.90 ioProgrammo con Libro (11 numeri + libro) €86,90 sconto 30% sul prezzo di copertina di €123.90 ioProgrammo Basic + Internet Magazine Go!OnLine(22 numeri + 22 CD-Rom) € 64,90 anziché € 129,80
ioProgrammo con Libro + Internet Magazine Go!OnLine(22 numeri + 22 CD-Rom + 11 Libri) € 89,90 anziché € 177,80)
Offerte valide fino al 30/10/04
ESTERO: Abbonamento Annuale: ioProgrammo Basic (11 numeri): € 151,80. ioProgrammo Plus (11 numeri + 6 libri): € 257,00
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Cesare Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- **cc/p n.16821878** o **vaglia postale** (inviando copia della ricevuta del versamento insieme alla richiesta);
- **assegno bancario non trasferibile** (da inviarsi in busta chiusa insieme alla richiesta);
- **carta di credito**, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- **bonifico bancario** intestato a Edizioni Master S.r.l. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master Servizio Clienti - Via Cesari Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel.02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelliera, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore- zona ASI - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Settembre 2004

EDIZIONI MASTER

Certificato n.5113 del 25/11/2003

L'Universo Tecnologico
www.itportal.it

Edizioni Master edita: Idea Web, Go!OnLine Internet Magazine, Win Magazine, PC Fun extreme, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, TV e Satellite, Win Extra, Home entertainment, Digital Japan, Digital Music, Horror Mania, ioProgrammo Extra, Le Collection.

News

LA CAMPAGNA ACQUISTI DI IBM

Era dal 1991 che in casa IBM non si vedeva una campagna di assunzioni di queste proporzioni: il 2004 si chiuderà con un totale di 18.800 nuovi assunti in tutto il mondo, l'88% in più rispetto alle pur rosee previsioni.
Un terzo dei nuovi impiegati è per il Nord America, mentre circa il 60% della forza lavoro sarà impiegato nei servizi di consulenza e nei nuovi campi aperti da Linux e dal Grid computing.
Alla fine dell'anno IBM avrà ben 330.000 dipendenti, confermandosi tra le primissime realtà dell'informatica mondiale.

www.ibm.com

LA LUNGA STRADA DEGLI LCD

La società di ricerca Meko ha messo in luce che, nonostante l'impetuosa crescita delle vendite degli LCD, le TV saranno ancora a lungo il regno incontrastato dei CRT. Il vantaggio di prestazioni di questa tecnologia vecchia ormai cento anni è il principale motivo di questa supremazia: ancora nel 2007, il rapporto fra CRT e LCD venduti sarà dell'ordine di tre a uno.
La tecnologia CRT potrà contare su significative evoluzione che, ad esempio, porteranno a breve una nuova linea di schermi CRT Philips dalla profondità enormemente ridotta.

GOOGLE FA CENTRO IN BORSA

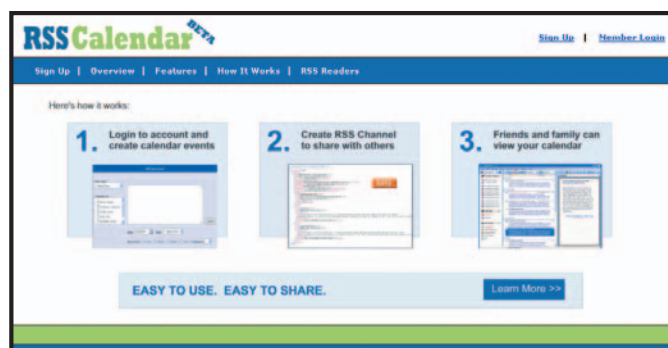
Alla fine è andata! Dopo un iniziale passo falso, le azioni di Google sono andate a ruba. Una errata valutazione (o era solo ingordigia?) dei fondatori li aveva spinti a fissare una forchetta di prezzo fra i 108 e i 135 dollari. Troppo. Le azioni sono infine state collocate a 85$, riducendo il valore iniziale dell'azienda di ben 10 miliardi di dollari. A quel prezzo, c'è stata una vera corsa all'acquisto che ha spinto il titolo a superare agevolmente i 100 dollari ad azione.
Anche se con qualche ombra, è innegabile il successo anche azionario di Google: 2,8 miliardi di dollari racimolati in poche ore. Cosa ci faranno con tutti questi soldi?
Questa è la domanda che gira nelle teste di tutti gli analisti e, soprattutto, nei vertici di Yahoo e Microsoft. Questi due giganti potrebbero essere le principali "vittime" di una ulteriore espansione di Google: Yahoo potrebbe soffrire della concorrenza spietata di G-mail, il cui successo ne ha addirittura preceduto il lancio. Microsoft, dal canto suo, potrebbe rischiare di essere messa in un angolo da un sistema operativo firmato Google che, invece che sui desktop, girerà direttamente on-line. Chi si ricorda delle teorie sui "PC leggeri" che avrebbero dovuto invaderci,

I FILM SU DISCHI OLOGRAFICI

Scienziati giapponesi sono alle prese con una nuova tecnica olografica in grado di memorizzare su supporto ottico una quantità di dati pari a duecento volte quella presente in un DVD a singolo strato.
L'impiego ideale è quella della riproduzione di film ad alta definizione, anche grazie alla velocità di trasferimento dei dati pari a 1 GB al secondo.

NUOVE APPLICAZIONI SU RSS

La tecnologia che ha permesso l'esplodere dei Blog comincia ad uscire dall'ambito dei diari on-line: il successo di un'applicazione di scheduling lascia prevedere un futuro ricco di novità sul fronte RSS.
John Pacchetti è lo sviluppatore di *RSSCalendar*, un'applicazione che consente agli utenti di convertire e pubblicare informazioni schedulate nel tempo. Insomma, la classica agenda on-line, condivisibile da gruppi di utenti, ha trovato forse la sua più leggera ed efficace interpretazione. Lo scopo dell'applicazione è proprio rendere più semplice un'operazione come quella della condivisione di appuntamenti che, al momento, richiede strumenti generalmente complessi e inutilmente ingombranti.

www.rsscalendar.com/rss/

utilizzando potenza di calcolo e storage disponibili in Rete? Già oggi, per le sue ricerche, Google può vantare un sistema di calcolo composto da migliaia di PC Linux collegati in Rete: cosa ne sarà dei S.O. Microsoft? Ovviamente la strada non è in discesa e sia Microsoft che Yahoo stanno prendendo tutte le contromisure per respingere gli attacchi di Google nei loro rispettivi settori. Inoltre, Microsoft e Yahoo sono impegnati nella ricerca di un algoritmo di ricerca che eguagli e superi l'attualmente imbattuto Google. È infatti nell'accuratezza delle sue ricerche che Google conserva il suo tesoro. Perdere terreno in quel campo potrebbe essere fatale per la nuova stella di Wall Street.

www.google.it

AMD ALLA CONQUISTA DEI TABLET

Sfruttando l'onda lunga di un buon momento per il mercato dei portatili, AMD ha lanciato due nuove versioni mobile dell'Athlon XP e dell'Athlon 64. Quest'ultimo è orientato ai notebbok ad alte prestazioni, garantendo una velocità di 2,2 GHz e integrando 1MB di cache. Più interessante l'Athlon XP-M2200+ che si inserisce nella battaglia dei notebook ultraleggeri, in cui AMD è da pochi mesi diventata protagonista. 1,6 GHz e 512 KB di cache sono le caratteristiche base. Il primo produttore ad approfittare di questo nuovo chip è stato Averatec, con il suo C3500: un notebook convertibile in tablet PC, leggero e con un ottimo sistema di risparmio energetico.

IBM VUOLE INSTALLAZIONI STANDARD

Il W3C sta vagliando una tecnologia proposta da IBM per unificare il packaging e le installazioni dei software su tutti i principali sistemi operativi: Solution Installation for Autonomic Computing. Il packaging sarà basato su XML e agevolerà la diffusione delle tecnologie on demand, compresa l'autoconfigurazione prevista dall'Autonomic Computing.

Accanto a Big Blue, troviamo nomi del calibro di Novell e InstallShield, tutti interessati a semplificare e rendere più robuste le operazioni di installazione e configurazione in ambienti eterogenei. Grandi risultati si aspettano anche nella fase di aggiornamento del software, fase che si dimostra sempre più costosa e delicata con le attuali tecnologie.

www.ibm.com/autonomic/

IL PRINCIPIO DI INDUZIONE

Le major americane sono impegnate in un pressing forsennato per far passare una legge che vieti il peer-to-peer e tutte le tecnologie che possano indurre a infrangere le leggi sul diritto d'autore. L'*Inducing Infringement of Copyrights Act* (abbreviato in *"Induce Act"*) comporterebbe conseguenze pesanti anche per tutti i prodotti legati alla riproduzione di mp3 come i lettori portatili. Dunque, non saranno solo gli utenti ad essere colpiti da questo nuovo giro di vite: tutti i produttori di tecnologie legate al file-shering e agli mp3 sono dunque in allerta, al punto che si è mossa la *Consumer Electronics Association* (CEA) per convincere il congresso a rivedere in modo meno restrittivo la normativa. La CEA riunisce oltre 1700 aziende americane impegnate nello sviluppo tecnologico, sviluppo che sarebbe messo a dura prova da una legge ceca e liberticida come l'Induce Act.
In realtà un campo di applicazione corretto per l'Induce Act ci sarebbe: perché l'America non si impegna proibire gli oggetti che inducono alla violenza e all'omicidio?

PARTITO UFFICIALMENTE IPV6

Icann ha annunciato l'adozione del nuovo protocollo nei server di root che controllano il traffico di Internet. L'IPv6 consente di indirizzare una quantità di dispositivi infinitamente maggiore rispetto agli attuali limiti e apre scenari inimmaginabili in cui miliardi di oggetti potranno essere presente in Rete e scambiare informazioni.
L'attuale IPv4, sulla breccia da quasi venti anni, mostrava ormai la corda, se si pensa ad esempio che quasi il 70% degli indirizzi risulta ormai occupato. Il nuovo IPv6 aumenta di tredici volte l'ordine di grandezza degli indirizzi utilizzabili, garantendo un lungo periodo di tranquillità agli organismi che regolano la Rete.
Per evitare possibili idiosincrasie, IPv4 resterà attivo parallelamente a IPv6 per i prossimi venti anni: c'è tutto il tempo per adeguarsi.

UN NUOVO REGALO PER L'OPEN SOURCE

IBM dona il codice sorgente del suo Database Cloudscape alla comunità Open Source. Sarà l'Apache Software Foundation ad essere il diretto destinatario di questa ingente donazione (circa mezzo milione di righe di codice). Big Blue ha affermato che continuerà a supportare i propri clienti e che si aspetta dalla comunità Open Source dei grossi miglioramenti. Il codice del database, 100% Java, potrà fornire una nuova spinta specialmente nel campo di applicazioni che utilizzino DB embedded.

IL SUCCESSO DELL'ITALIAN PERL WORKSHOP

Il 19 e 20 luglio 2004 si è svolto nella sede dell'Università di Pisa il primo workshop nazionale degli utenti italiani del linguaggio Perl, organizzato da Perl.it (*www.perl.it*) e dal gruppo Perl Mongers di Pisa (*http://pisa.pm. org*). I talk, interessanti e pratici, sono stati molto apprezzati dagli oltre numerosi iscritti. Fra gli argomenti affrontati: Unicode, XML, SOAP, accesso ai database, sviluppo di applicazioni web, sistemi di content management, integrazione con applicativi di produttività (OpenOffice, Microsoft Excel), interfacce grafiche, CPAN e qualche anticipazione di quello che sarà Perl 6. Dall'inizio di ottobre verranno inoltre pubblicate su *www.perl.it* le slide degli interventi con cadenza settimanale. La manifestazione si è conclusa con un simpatico 'social event', ossia un'asta nella quale sono stati 'battuti' materiali (libri e t-shirt) donati

da O'Reilly, sponsor della manifestazione. L'asta è stata organizzata per la raccolta di fondi che saranno destinati alla Perl Foundation. L'Italian Perl Workshop è stato reso possibile grazie anche alle sponsorship con Activestate, KSolutions, Linux&C, Leader.IT S.r.l., PC Express S.r.l. e con il patrocinio di Ulis S.r.l. I saluti finali sono stati un arrivederci alla prossima edizione dell'Italian Perl Workshop nel 2005 e soprattutto all'"Italian Code Jam", un evento di portata internazionale dedicato al Software Libero e all'Open Source che si terrà presso l'Università di Ferrara il 9 ottobre 2004. Organizzato da ERLUG, FerraraLUG, RELUG, PiacenzaLUG, ParmaLUG, vede anche Perl.it tra gli organizzatori. Tra gli ospiti, Larry Wall in persona (il creatore del linguaggio Perl), Dave Cross (coordinatore dei gruppi Perl Mongers per la Perl Foundation) e Allison Randal (project manager di Perl 6), oltre a personaggi di spicco dell'Open-Source come Andrea Arcangeli (kernel linux developer), Moshe Bar (openmosix), Alex Martelli (Python developer) ed altri ancora.

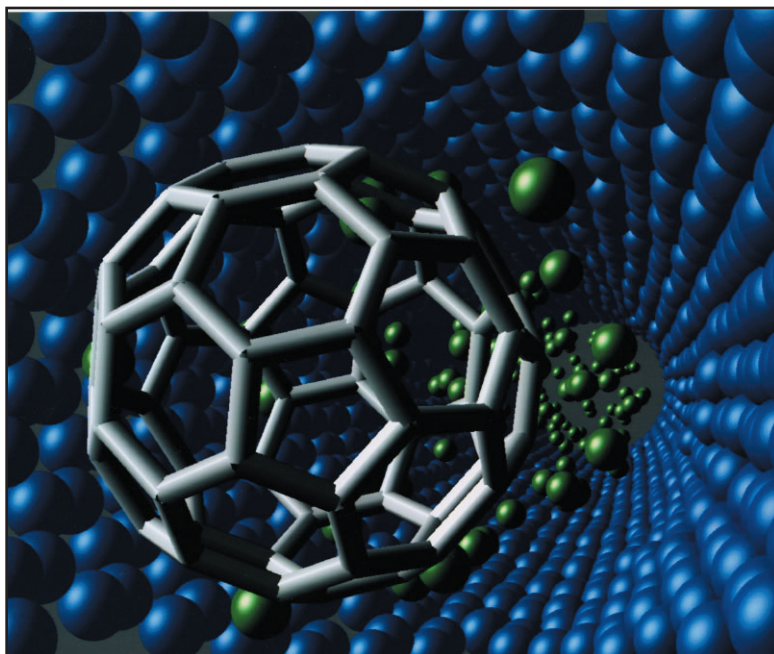
www.perl.it

FIAT E MICROSOFT PER UN'AUTO INTERATTIVA

Le due aziende hanno siglato un accordo di collaborazione di lungo periodo per una partnership tecnologica. L'obiettivo è di sviluppare sistemi telematici in grado di connettere al meglio gli automobilisti con il mondo esterno. Già dal prossimo anno, le vetture del gruppo torinese potranno avere a bordo il Windows Automotive, una tecnologia che consentirà di integrare, via Bluetooth, PDA e cellulari con il computer di bordo. Agenda e rubrica personali saranno sempre disponibili al guidatore che, attraverso un'interfaccia comune, potrà accedere ad un ricco set di servizi, incluso il celebre sistema di navigazione CONNECT della Fiat. Tra le funzioni supportate, ci sarà anche un collegamento USB che permetterà di ascoltare la musica immagazzinata in qualsiasi dispositivo portatile.

LE NANOTECNOLIGIE SFRUTTERANNO LA FORZA DELLA NATURA

Alla conferenza Hot Chips, tenutasi alla Sanford University nell'agosto scorso, è emersa una interessante tendenza per le nanotecnologie: l'utilizzo o l'imitazione di organismi che si trovano in natura. Steve Jurvetson, da anni impegnato in questo tipo di ricerche, ha portato una serie di esempi di questa interazione: si va da un microbo capace di produrre un metallo utile per la costruzione di microchip, a memorie capaci di immagazzinare dati utilizzando molecole al posto di transistor, molecole progettate imitando la clorofilla.
Assieme alle note positive, emergono anche degli inquietanti interrogativi: l'imitazione del comportamento naturale porterà all'autoriproduzione. Il risparmio in termini produttivi sarebbe enorme, resterebbero da verificare gli effetti dal punto di vista ecologico: inquinamento da nanotecnologie? E se non si riuscisse a fermare la riproduzione di questi microbi artificiali?

www.hotchips.org

LONGHORN "LIGHT" NEL 2006

A sorpresa, Microsoft ha annunciato che Longhorn sarà disponibile entro la fine del 2006. Quando tutti, Gates compreso, erano convinti che il nuovo Sistema Operativo difficilmente sarebbe stato lanciato prima del 2008, un cambio di strategia ha permesso questo scatto: la chiave di volta è stata la rinuncia la rivoluzionario File System che avrebbe dovuto essere integrato il Longhorn. WinFS, con le sue innovative capacità di memorizzare e ricercare i file come in un database ad oggetti, sarà disponibile solo in un secondo momento. L'attuale scheduling prevede che WinFS sarà rilasciato in versione beta, nello stesso momento in cui sarà disponibile la prima versione ufficiale di Longhorn. Benché pesante, la rinuncia al WinFS ha permesso un'accelerazione fondamentale ai piani di sviluppo del nuovo sistema operativo che avrà comunque molte innovazioni da sfoggiare: il nuovo motore grafico vettoriale Avalon, ed il sistema di connessione Indigo resteranno come fiore all'occhiello di questa versione "depotenziata" di Longhorn. Queste due ultime tecnologie rientrano nella piattaforma WinFX che permetterà di sviluppare applicazioni per Longhorn: altro importante annuncio, specialmente per gli sviluppatori, è che proprio WinFX sarà disponibile, come aggiornamento, per Windows XP w Windows Server 2003. Insomma, il colpo è stato grande, ma Jim Allchin invita a non chiamarlo Shorthorn, le novità ci saranno è saranno in gran numero: resta la promessa mancata di avere un file system che consenta di semplificare l'accesso alla immane quantità di materiale multimediale presente negli odierni PC. I due anni di ritardo preventivati peri il rilascio di WinFS non sono poca cosa, specialmente se si pensa che WinFS dovrebbe essere la carta che Microsoft vorrebbe giocare per vincere la battaglia con Google sui motori di ricerca.

WinFS
Models
APIs
Objects | T/SQL | XML
Framework
Services
Synchronization (WinFS, Win32, ...)
InfoAgent (PreferenceRules, ...)
Schemas
People | Groups | Calendar | Media | Documents | ...
Core WinFS
Operations
Filesystem Services (Metadata Handlers, ...)
Data Model
Items | Relationships | Extensions
Relational Engine
Kernel Mode
Identity & Security System
Transactions
Storage
Distributed File System
Volume Shadow Copy Service (Data Protection)
File Replication Service
Virtual Disk Service
Client Side Caching | Redirectors | Filter Manager | Cache Manager
Transactional NTFS | Universal Data Format | CDFS | FAT 16/32
Kernel | Mt Rainer CD/DVD/Blue | Serial ATA/ Serial SCSI | 1394B/USB SD/CF | iSCSI/ SAN | ...
CLR
Serialization
IO Manager
Bandwidth Managed I/O

Updater 2 per Macromedia Flash MX 2004

Macromedia Flash MX 2004 Versione 7.2

Il nuovo update gratuito per Flash MX 2004 risolve alcuni problemi e introduce anche nuove funzionalità che arricchiscono e rendono sempre più completo il gioiello di casa Macromedia

di Marco Casario

Siamo giunti al secondo aggiornamento per Flash MX 2004. Il primo uscì nel lontano Novembre 2003 e risolveva molti dei piccoli problemi che la nuova versione del programma portava con sè.
Sin dal primo lancio del programma Macromedia è sempre rimasta con l'orecchio teso ad ascoltare i propri clienti per cercare di soddisfare le loro esigenze, e questo nuovo update ne è la conferma. Anche se ancora non sono molto lontani i tempi in cui Flash veniva usato esclusivamente per creare accattivanti quanto inutili introduzioni animate per siti web, la comunità di sviluppatori Macromedia sta metabolizzando la vera nuova essenza del programma, quella di un potente ambiente di sviluppo per la creazione di complesse applicazioni capaci di una ricca interattività.
Fare un elenco dettagliato di tutti i bug risolti (e di quelli ancora da risolvere ;)) nel nuovo update è, oltre che impossibile, anche al di fuori degli obiettivi di questa breve recensione; quello che invece è interessante capire è su cosa vertono, in pratica e come questi bug-fix (risoluzione dei problemi) migliorano l'uso del programma da parte degli sviluppatori. Andiamo per questo a dettagliare ognuno dei tre punti elencati, aggiungendo informazioni preziose.

Fig. 1: **La nuova Start Page, più veloce e meno avida di risorse**

AUMENTO DELLE PERFORMANCE E USO DELLE RISORSE

È sicuramente uno degli argomenti che possono rendere frustrante l'uso del software per uno sviluppatore. Mentre infatti è possibile aggirare un problema tecnico, derivato da un non corretto funzionamento del programma, mediante soluzioni più o meno eleganti, l'impotenza da parte degli utilizzatori di fronte all'inesauribile tempo di compilazione di un file è veramente devastante. Questo aspetto, quelli di Macromedia lo hanno recepito grazie alle lamentele della comunità degli sviluppatori Flash. Ecco quindi che, oltre alle performance e al resource usage, ci sono migliorie sui tempi di compilazione e pubblicazione di file, una gestione più fluida e performante dei progetti che contengono molte righe di codice, tempi di attesa per lo start up del programma ridotti al minimo, aumenti nella stabilità del programma su sistemi Windows con minori risorse di sistema e consumi della CPU minimizzati sulla Start Page. La Start Page è una delle nuove caratteristiche della suite Macromedia MX Studio 2004.
In FlashMX 2004 questa schermata creava dei problemi quando il programma era in idle, cosumando molte risorse della CPU. Nel nuovo update questo bug è stato risolto. Con tutti questi problemi risolti sono terminate (incrociamo le dita) le giornate in cui si doveva interrompere l'esecuzione del programma e riavviarlo per esportare un progetto o correre presso il negozio di fiducia per acquistare un nuovo banco di RAM al fine di rendere il programma più veloce e performante!

DOCUMENTAZIONE

Dico la verità. Anche per un amante di Flash come me, è stato un duro colpo scoprire che la politica di Macromedia sarebbe stata col tempo quella di portare tutta la documentazione cartacea in formato elettronico (questo passaggio è per la precisione avvenuto tra la versione di Flash 5 e la Flash MX). Aprire la confezione di Flash MX e non trovare i manuali non è stato facile da digerire, ma tant'è: bisogna accontentarsi della versione elettronica della documentazione con tutti gli svantaggi (ed i vantaggi) connessi. La documentazione, nonostante sia molto voluminosa, non riesce a coprire tutti gli aspetti del programma. Molte volte vi sarete trovati nella situazione in cui un determinato task o una determinata funzione non era

documentata a dovere. Uno degli scopi di questo update era proprio quello di fornire una documentazione aggiornata, migliorata dal punto di vista tecnico e che risolvesse anche alcuni bug riscontrati.
Quelli di Macromedia dichiarano di aver risolto oltre 2000 bug relativi alla sola documentazione e oltre 150 sugli esempi di codice che essa riportava. Sono state inoltre aggiunte nuove sezioni, come quella di Best Practise per un corretto utilizzo del programma e di approccio allo sviluppo, una nuova finestra "Welcome Page" per aiutare i nuovi utenti nella ricerca di informazioni in maniera più veloce ed intuitiva, è stata riscritta la parte sulla creazione di Components e, per la serie Last but not Least, sono stati aggiunti 14 nuovi esempi (file FLA con codice aperto) sotto la cartella di sistema *C:\Programmi\Macromedia \FlashMX2004\Samples*.Insomma veramente lodevole il lavoro dei Technical Writers.

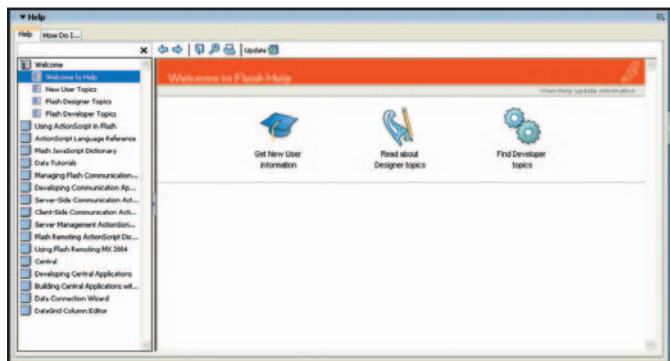



Fig. 2: **La Welcome Page per aiutare gli utenti nella ricerca di informazioni**

La Welcome Page si trova all'interno del pannello Help di Flash. È una semplice ma molto intuitiva pagina che accompagna gli utenti meno esperti in un percorso didattico guidato e pensato per la ricerca e quindi l'acquisizione veloce delle informazioni sui vari aspetti del programma.
È infatti divisa in tre sezioni:

1. **Get New User information** (comprende le *Quick Start Lessons* e i *New User Topics*)
2. **Read about Designer topics** (comprende le *Basic Flash Lessons*, il *creating cotents*, e le guide sul designing in flash)
3. **Find Developer topics** (comprende le *Basic Actionscript lessons*, la guida all'*Actionscript language* e il *Working with Actionscript*).

COMPONENTI V2

La nuova versione di Flash Mx 2004 implementava un nuovo modello di architettura per i componenti, denominato appunto V2 (che sta per versione 2). Il nuovo modello offre agli sviluppatori un'architettura robusta per creare componenti riusabili e dalla ricca interfaccia utente. Ci sono molti articoli dedicati a questo argomento e per approfondire consiglio di leggere e seguire i tutorial indicati nel box laterale, ma quello che ci interessa analizzare in questo articolo riguarda i problemi risolti con questo update per Flash relativi ai nuovi componenti. Non riesco più a prolungare l'attesa di dare questa notizia: finalmente è stato di nuovo aggiunto alla lista degli UI Components (User Interface) lo Scrollbar component (inspiegabilmente tolto nella versione 2004). Sono stati aggiornati anche i Media Component per l'importazione e la gestione dei video e di file mp3, risolti dei problemi sul Datagrid component, sulla TextArea e sul Combobx component e dal punto di vista dell'IDE del programma è stato fissato il problema relativo al pannello Components che non "ricordava" il suo stato e apriva per default tutte le categorie di componenti quando veniva ricaricato. Bug molto noioso per chi aveva installato un numero elevato di oggetti. Continuando con i bug molto noiosi, il Flash Team non poteva far finta di nulla riguardo il problema dei componenti caricati all'interno di altri swf. Infatti, importando un swf al cui interno erano presenti dei componenti, questi ultimi non funzionavano correttamente. La soluzione temporanea a questo problema ci era stata data proprio da Macromedia nelle Technotes di Flash e consigliava di usare due metodi : caricare il movie con il componente in un livello invece che in un taget movieclip oppure caricarlo in un tagert movieclip con la funzione *loadMovie("url",target)* e inserendo sullo stage un'istanza del componente (la technote completa la leggete qui: *www.macromedia.com/support/flash_remoting/ts/documents/combobox.htm*). Da non dimenticare inoltre che a luglio sono stati rilasciati anche i nuovi Flash Remoting 2004 Components, riscritti interamente in Actioscript 2 e che comprendono il Flash Remoting Connector. Sono scaricabili al seguente indirizzo: *www.macromedia.com/software/flashremoting/downloads/components/*

MAMMA HO PERSO LE CARTELLE!

Non sono impazzito in chiusura di articolo, ma volevo solo attirare l'attenzione di voi lettori per mettervi al corrente che tra i cambiamenti fatti in questo update, sono state anche cambiate le disposizioni di alcune cartelle del programma:

- i componenti che si trovano nelle cartelle UI Components, Data Components, e Media Components all'interno della cartella FirstRun saranno spostati al livello di configurazione dell'applicazione (*C:\Programmi\Macromedia\Flash MX 2004\en\Configuration\Components*), mentre quelli che si trovano al livello di configurazione utente verranno salvati;
- la cartella *Importers* sarà spostata dalla cartella *First Run* al livello di configurazione dell'applicazione;
- la cartella *Libraries* e *Templates* sarà spostata dalla cartella *First Run* al livello di configurazione dell'applicazione;
- la cartella *Samples* sarà spostata dalla cartella *First Run* al livello di configurazione dell'applicazione;
- la cartella *ComponentFLA* sarà spostata dalla cartella *First Run* al livello di configurazione dell'applicazione.

☑ Upgrade Flash MX 2004 version 7.2
Produttore: Macromedia
Sul web: www.macromedia.com
Prezzo: Gratuito

SOFTWARE SUL CD

Eclipse 3.0

Il massimo per sviluppare in Java

Il progetto Eclipse, inizialmente di proprietà IBM e da questa donato alla comunità Open-Source, è essenzialmente una raccolta di vari tipi di sottoprogetti che in realtà insieme formano, più che un software per lo sviluppo, un framework per la creazione di ambienti integrati di sviluppo per qualunque linguaggio e piattaforma. Ma la ragione del successo di Eclipse risiede nell'IDE per la creazione di applicazioni Java che, offerto di fatto come esempio di uso del framework che Eclipse mette a disposizione, è riuscito a imporsi come prodotto di alta qualità, pur essendo completamente gratuito.

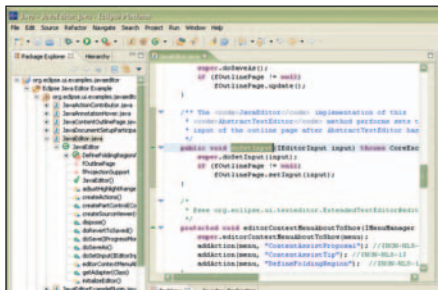



Fig. 1: **L'interfaccia, davvero completa, in un primo momento può disorientare. Presto ci si sente viziati da tante comodità!**

L'INSTALLAZIONE

L'installazione del prodotto è veramente molto semplice. Dovete assicurarvi che sia installato sulla vostra macchina un JDK pari o superiore al 1.4.1, dopodichè potete decomprimere il file direttamente dove volete installare il prodotto finito.
Tra i vari file estratti troverete l'eseguibile *eclipse.exe* da lanciare per avviare l'IDE. Nella cartella presente nel CD trovate alcuni plug-in necessari a seguire la sezione dedicata alla programmazione visuale all'interno del corso su Eclipse contenuto nella rivista.

IL PRIMO AVVIO

Dopo la richiesta della directory del workspace, dove verranno tenuti i vostri progetti (lasciate pure il default), vi accoglierà la schermata di benvenuto con vari link ad elementi di aiuto e supporto per chi è nuovo all'ambiente. Chiudendo la finestra Welcome vi ritroverete invece nell'IDE vero e proprio. Eclipse tenta di agevolare il lavoro del programmatore organizzando i vari elementi offerti in prospettive (*perspective*): ogni prospettiva è un insieme di viste (*view*), cioè di finestre che offrono le più svariate funzionalità (explorer del repository di progetti, outline delle classi sviluppate, elenco dei package di un progetto, gerarchia delle classi, problemi di compilazione, etc), permettendo così al programmatore di lavorare secondo le sue esigenze: la prospettiva Java per scrivere il codice, quella Debug per fare il debug delle applicazioni, quella dedicata al team *synchronising* per lavorare con repository CVS, e via dicendo. Una piccola toolbar in alto a destra della finestra dell'IDE vi consente di aprire le varie prospettive e di passare tra una e l'altra.

L'INTERFACCIA

Come buona parte degli ambienti di sviluppo disponibili, Eclipse propone un editor integrato con syntax highlighting completamente configurabile (menu *Window->Preferences*, poi *Java->Editor,* pagina Syntax, come in Fig. 1) ed una comodissima funzione

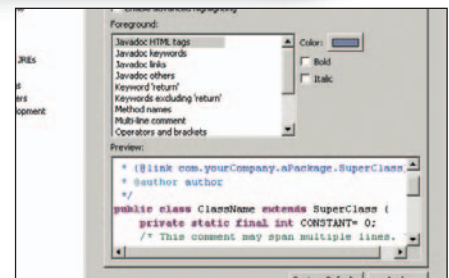



Fig. 2: **Le opzioni di syntax colouring offerte da Eclipse per l'editing Java**

di formattazione del codice, anch'essa completamente configurabile con un numero di opzioni impressionante (menu *Window-> Preferences*, poi *Java->Code Style->Code Formatter,* create un profilo personalizzato e cliccate su Edit, come in Fig. 2) che prende tutto quello che avete scritto e lo sistema in base alle vostre scelte e alle convenzioni di stesura del codice previste da Java: basta premere *CTRL+ SHIFT+F* durante l'editazione (mette in ordine indentazioni, spaziature, ritorni a capo, tutto insomma: è davvero ben fatto ed è una gran cosa per chi, come me, ci tiene ad avere il codice scritto in maniera pulita e leggibile). Un'altra comodissima ed innovativa caratteristica di Eclipse è quella di mostrare sempre tutti gli eventuali problemi di compilazione del vostro codice nella *view* dei problemi (*problems view*), grazie alla compilazione automatica delle classi che viene fatta ad ogni salvataggio del vostro lavoro (con *CTRL+S*). In questa maniera, ogni volta che salvate avete un immediato feedback sulla correttezza di quanto digitato: è quindi una buona idea abituarsi a salvare molto spesso, oltre che per evitare spiacevoli perdite qualora venisse a mancare la corrente, anche per verificare subito la reazione del compilatore a quello che abbiamo scritto.

☑ Eclipse 3.0
Produttore: consorzio eclipse.org
Sul web: www.eclipse.org
Licenza: CPL (Common Public License)
Nel CD: *Eclipse*

HexToolbox 2.2

Per manipolare file di grandi dimensioni

Un potente editor esadecimale che consente di visualizzare e modificare file di grandi dimensioni (fino a 4 GB). Oltre all'ASCII supporta la codifica EBCDIC tipica dei mainframe. Consente operazioni di drag&drop e spicca ottime prestazioni nella ricerca raggiungendo il ragguardevole traguardo di 1GB al minuto su un PC di medie prestazioni. Tutto è impostato per lavorare al meglio con file di grandi dimensioni. Ad esempio, molto interessante è la ricerca in background: si lancia la ricerca e si può subito continuare a lavorare sul file, senza aspettare l'esito della ricerca. L'interfaccia ha una impostazione classica ed è decisamente intuitiva, anche se avremmo preferito poter variare dimensione e tipo di caratteri che non si presentano molto leggibili. Aprendo più finestre, è possibile lavorare su più file contemporaneamente. Non mancano le utility indispensabili per chi lavora in esadecimale: convertitore di base e calcolatrice scientifica sempre a portata di mano. Insomma, uno strumento di indubbia efficacia per tutti coloro i quali sono curiosi di esplorare i contenuti di file eseguibili per carpirne i segreti o per modificarne qualche particolare. Da usare con cautela, stante l'alto rischio di danneggiare irreparabilmente qualsiasi applicazione. Versione di prova valida trenta giorni. HexToolbox continua a funzionare al 100% anche dopo la scadenza dei trenta giorni: l'unico fastidio è un pop-up che ci invita a registrare il prodotto. Molto civile.

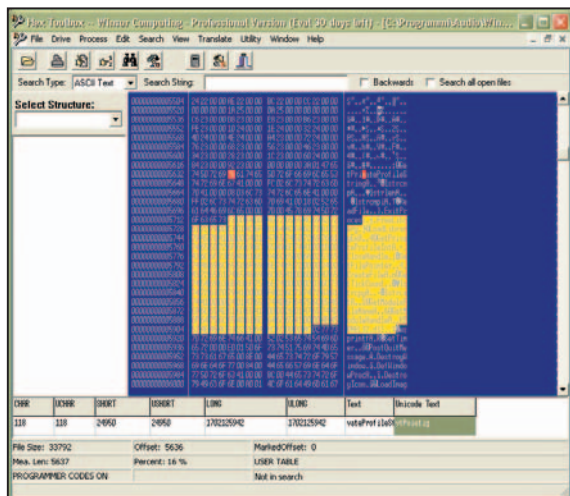



Fig. 1: L'interfaccia di HexToolbox, semplice e molto intuitiva

☑ HexToolbox 2.2
Produttore: Winsor Computing
Sul Web: www.winsorcomputing.com
Prezzo: $29.95
Nel CD: *HexToolboxSetup.exe*

Exchanger XML Editor 2.0

Creare e gestire documenti XML

Un editor XML Java-based che offre un ampio spettro di funzionalità, sia per chi debba interagire solo con i dati contenuti in un documento XML sia per gli sviluppatori. Molto ricca la scelta fra le visualizzazioni disponibili: vista ad albero, schema-based e l'ottima tag-free per inserire e visualizzare dati in modo più chiaro. Effettua la validazione dei documenti e, grazie alle regular expression, è possibile fare delle ricerche molto dettagliate.
Ottimo il supporto per XSLT 2.0 e per le trasformazioni XSLFO. Altri standard supportati dall'editor sono RelaxNG, XPath, XQuery e SVG. Da rimarcare la presenza di un ottimo debugger XSLT che si occupa di mostrare a video, contemporaneamente, tutti i file coinvolti nella trasformazione XSLT: sia l'XML di partenza sia tutti quelli risultanti dalla trasformazione. Tutte le variabili relative alla trasformazione sono riportate in un apposito pannello di dettaglio sulla destra dello schermo. La navigazione all'interno dei documenti XML è significativamente semplificata dall'efficace integrazione delle ricerche XPath: i risultati delle query sono riportati nella finestra in basso e, attraverso un semplice click, viene evidenziata la porzione di XML relativa al risultato indicato. Al primo avvio, è necessario richiedere una chiave di attivazione, specificando nome e indirizzo mail. In pochi istanti, riceverete file xml da copiare nella directory di installazione. Da questo momento avrete trenta giorni di tempo per provare l'applicazione.

Fig. 1: Visualizzazione con vista ad albero

Fig. 1: La vista tag-free consente di inserire dati in modo semplice

☑ Exchanger XML Editor 2.0
Produttore: Cladonia Ltd
Sul Web: www.exchangerxml.com
Prezzo: €85.00
Nel CD: *xngrV20_windows_jvm.exe*

Apache 2.0.5

Affidabilità, sicurezza e stabilità: gli ingredienti del suo successo

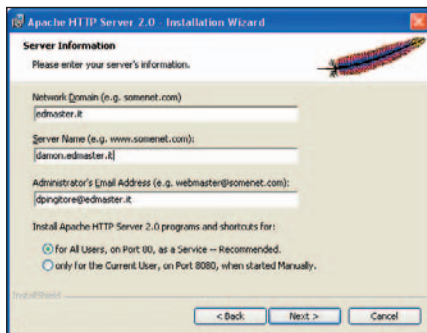
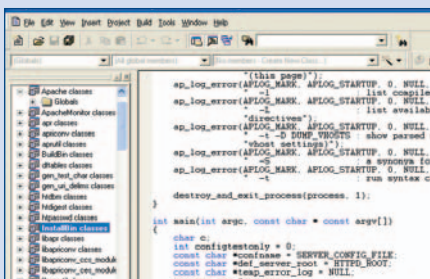
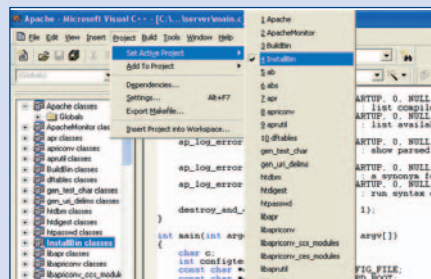
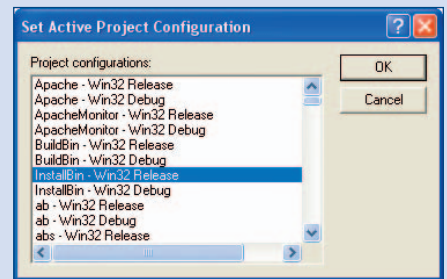
Apache è un server Web creato e distribuito con licenza GPL dalla Apache Software Foundation. L'obiettivo di questo progetto è fornire un server Web sicuro, efficiente ed estendibile che fornisca servizi HTTP affidabili e nel pieno rispetto degli standard. Apache è il Web server più popolare su Internet ormai dal "lontano" Aprile 1996. Secondo un'indagine condotta da Netcraft, nell'Ottobre 2003 più del 65% dei server presenti su Internet utilizzavano Apache. Le novità più rilevanti sono state introdotte con il passaggio dalla versione 1.3 alla attuale 2.0, mentre i rilasci più recenti, non includono novità di rilievo e sono soprattutto versioni di bug-fix.
Tra le novità più importanti introdotte con la versione 2.0 sono presenti il supporto per il protocollo IPv6, una nuova versione delle API e supporto per le piattaforme non Unix (BeOS, OS/2 e Windows) notevolmente migliorato.

Fig. 1: **Configurazione durante la fase di installazione**

INSTALLAZIONE E CONFIGURAZIONE DI BASE

La piattaforma predefinita per l'installazione di Apache 2.0 è NT e tutte le versioni di Windows che utilizzano il suo kernel come 2000 e XP. In realtà può essere utilizzato anche con le versioni consumer di Windows come 98, 98SE, ME e perfino 95, anche se questa pratica è fortemente sconsigliata. Il software è disponibile in tre formati differenti: Microsoft Installer (*.msi*), eseguibile (*.exe*) e sottoforma di codice sorgente in formato *.zip*. In quest'ultimo caso, prima di procedere con l'installazione vera e propria, è necessario decomprimere e successivamente compilare il codice sorgente per creare l'eseguibile. Per fare ciò è necessario utilizzare Microsoft Visual C++. L'installazione mediante Microsoft Installer è a dir poco banale, basta semplicemente eseguire il file e seguire le istruzioni. Durante questa fase è necessario fornire il nome del dominio DNS (es. *edmaster.it*), il nome del server (es. *damon.edmaster.it*) e un indirizzo e-mail per l'amministratore del server (per l'opzione in fondo alla schermata, è sufficiente utilizzare quella predefinita). Per il tipo di installazione selezionate Typical, mentre come directory di installazione utilizzate quella di default. Terminato il processo di installazione il server verrà avviato automaticamente e nella barra di stato di Windows sarà presente l'icona dell'Apache Monitor che consente di effettuare diverse operazioni tra cui l'arresto e l'avvio del server. Per verificare l'installazione di Apache basta semplicemente eseguire un comune browser e puntarlo all'indirizzo *http://localhost* o *http://127. 0.0.1* (indirizzo di loopback). Se tutto è andato per il verso giusto apparirà la pagina di benvenuto.

Fig. 2: **Verifica dell'installazione di Apache 2.0**

☑ Apache 2.0
Autore: Apache Software Foundation
Sul Web: www.apache.org
Licenza: Apache License
Nel CD: apache_2.0.zip

COMPILARE I SORGENTI DI APACHE PER WINDOWS

1 **Avviate Visual C++, dal menu *File* selezionate *Open Workspace...* e scegliete il file *Apache.dsw* presente nella cartella dei sorgenti.**

2 **Dal menu *Project/Set Active Project* selezionate la voce *InstallBin.* Questo assicura che successivamente tutti i progetti vengano compilati.**

3 **Selezionate *Build/Set Active Configuration...* successivamente scegliete *InstallBin – Win32 Release,* dopodichè *Build/Build Apache.exe.***

PHP 5.0

Non un semplice linguaggio di scripting ma una vera piattaforma di sviluppo

Il team di sviluppo di PHP ha recentemente rilasciato la versione definitiva del linguaggio di scripting PHP 5. Il pacchetto contiene numerosi aggiornamenti e molte novità assolute, che coprono diversi aspetti del linguaggio. Alcune delle nuove caratteristiche includono: il nuovo motore Zend Engine II, dotato di un nuovo modello ad oggetti e decine di nuove caratteristiche; il supporto XML completamente riprogettato, tutto realizzato attorno alla formidabile libreria *libxml2*; una nuova estensione di SimpleXML che facilità l'accesso e la gestione di oggetti XML e PHP; estensione integrata di SOAP per l'interoperabilità con i Web Services; una nuova estensione di MySQL, chiamata MySQLi, per gli sviluppatori che utilizzano MySQL 4.1 o versioni più recenti (questa nuova estensione include un'interfaccia orientata agli oggetti accanto a quella tradizionale. Inoltre PHP 5 integra al suo interno una versione completa del database SQLite.

Fig. 2: **Home page di Zend Technologies, azienda leader nello sviluppo di soluzioni PHP**

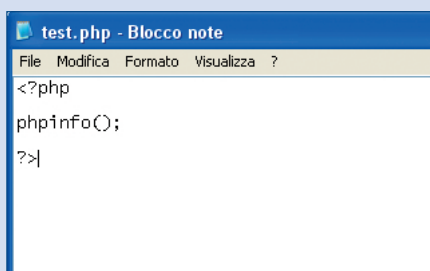
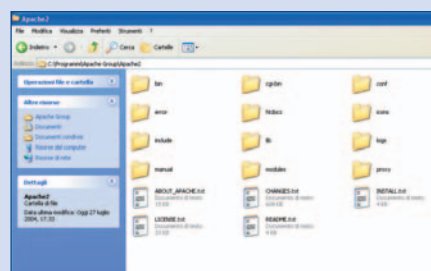
INSTALLARE E CONFIGURARE PHP 5

Sui sistemi MS Windows è possibile installare PHP in due diversi modi: mediante l'installer InstallShield oppure manualmente. Nel primo caso l'installer configura automaticamente anche i server Web IIS, PWS e Xitami ma non Apache. L'installazione manuale è leggermente più complessa ma consente di configurare PHP per l'utilizzo con qualsiasi server. Per prima cosa è necessario creare la directory *C:\PHP* (va bene qualsiasi posizione all'interno del filesystem, è stata scelta questa per comodità). Decomprimete il contenuto del pacchetto *.zip* all'interno della directory appena creata. Individuate (all'interno di C:\PHP) il file php *.ini-dist*, rinominatelo in *php.ini* e copiatelo nella directory *C:\WINDOWS*. Aprite il file con un comune editor di testi (Notepad), individuate il record *extension_dir* e inserite il valore C:\PHP, per indicare la directory di installazione di PHP, al cui interno sono posizionati gli eseguibili e le librerie. Allo stesso modo individuate la voce *doc_root* e inserite il percorso della *DocumentRoot* di Apache (la *DocumentRoot* è la posizione in cui vanno salvati gli script PHP creati che il server dovrà caricare). A questo punto prima di eseguire gli script, è necessario configurare Apache affinché utilizzi il modulo PHP. Per fare ciò è necessario editare il file di configurazione di Apache *httpd.conf*, posizionato all'interno della directory di installazione del server nella cartella *conf*. Aprite il file e aggiungete le righe seguenti:

Fig. 1: **Home page del progetto PHP**

```
LoadModule php5_module c:/php/php5apache.dll
AddType application/x-httpd-php .php
```

PHP 5.0
Autore: PHP Group
Sul Web: www.php.net
Licenza: PHP License
Nel CD: php_5.0.zip

COME VERIFICARE L'INSTALLAZIONE DI PHP

1 Create un nuovo file con *Notepad*, inserite i tag PHP di apertura e chiusura *<?php ?>*, e all'interno di questi la riga *phpinfo();*.

2 Salvate il file con il nome *test.php* all'interno della *DocumentRoot* di default (*"C:/Programmi/Apache Group/Apache2/htdocs"*).

3 A questo punto per eseguire lo script, basta semplicemente puntare un qualunque browser all'indirizzo http://localhost/test.php

MySQL 4.0.20

Il database server SQL Open Source stabile e veloce

MySQL è diventato in breve tempo il database server Open Source più popolare, con il tasso di crescita più elevato proprio all'interno delle industrie. Questo è dovuto alla volontà degli sviluppatori di fornire un prodotto efficiente e meno complicato dei rivali, che garantisce un TCO notevolmente ridotto. MySQL offre numerosi vantaggi: affidabilità e prestazioni elevate prima di ogni altra cosa, mentre la sicurezza è sempre di prim'ordine, grazie al lavoro di test svolto dalla comunità Open Source. Esistono numerose versioni del database server: Max DB, cluster e a breve una versione embedded.
Inoltre, sono disponibili tutta una serie di strumenti che migliorano l'efficienza e la produttività: *MySQL Control Center (MySQLCC)* un client grafico per la gestione dei dati; *MySQL Administrator* un'interfaccia visuale per l'amministrazione del server MySQL; *MySQL Connector/J* e *MySQL Connector/ ODBC* per la connessione attraverso driver JDBC (Java) e ODBC.

INSTALLAZIONE E STRUMENTI DI GESTIONE

I pacchetti di installazione di MySQL (la versione stabile è la 4.0.20) sono disponibili sottoforma di archivio *.zip*, ma prevedono due metodi di installazione differenti. Infatti, è disponibile un archivio che contiene solo i file che compongono la distribuzione MySQL, e un secondo archivio zip che comprende anche l'installer. Nel primo caso bisogna semplicemente decomprimere il file all'interno della directory *C:\mysql,* nel secondo, dopo aver scompattato il file in una cartella qualsiasi, basta accedere a questa directory ed eseguire il file *SETUP.EXE.* In quest'ultimo caso l'installazione è completamente automatizzata, bisogna solo indicare la directory di installazione *C:\mysql.*

Fig. 1: **Interfaccia del tool di gestione WinMySQLAdmin**

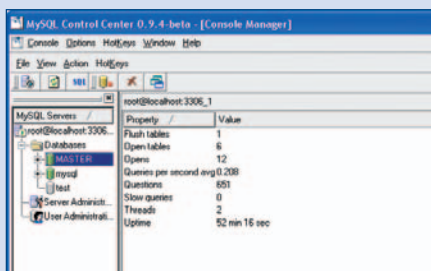
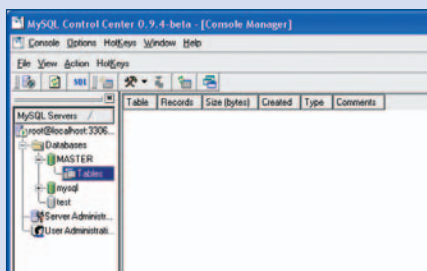
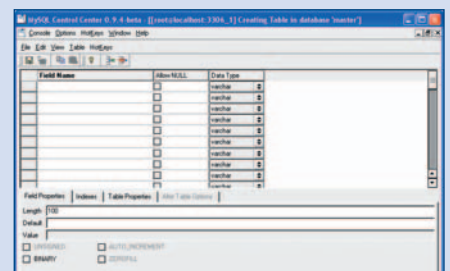
WINMYSQLADMIN

La versione per Windows di MySQL include un comodo tool dotato di interfaccia grafica che consente di gestire facilmente il sever.
Lo strumento in questione si chiama *WinMySQLAdmin* e l'eseguibile si trova nella directory *C:\mysql\bin.* Utilizzando questo programma è possibile avviare, arrestare il server, monitorarne l'attività, raccogliere informazioni sullo stato generale del servizio, visualizzare ed editare i file di configurazione, creare report e tanto altro.

MYSQL CONTROL CENTER

La struttura dell'interfaccia è in perfetto stile *Esplora* risorse di Windows, suddivisa in tre aree separate: sulla sinistra è presente la barra che visualizza le connessioni attive, in basso è presente la sezione che visualizza i messaggi relativi alle operazioni svolte, mentre la finestra principale consente di visualizzare e gestire tutte le informazioni contenute nel database.
Il vantaggio principale offerto da MySQLCC consiste senza ombra di dubbio nella maggiore velocità di esecuzione rispetto ad altri client grafici basati su interfacce Web realizzate in PHP e altri linguaggi di scripting. Inoltre il programma dispone di comandi per personalizzare il look and feel dell'interfaccia.

☑ MySQL 4.0
Autore: MySQL AB
Sul Web: www.mysql.com
Licenza: GNU GPL
Nel CD: mysql_4.0.zip

CREARE UN NUOVO DATABASE CON MYSQLCC

1 **Selezionate *Databases*, sulla toolbar premete il pulsante *New Database,* nella finestra di dialogo inserite il nome del database *(master).***

2 **Sulla sinistra selezionate *master*, sulla toolbar apparirà il pulsante *Connect* premetelo, in questo modo il database è pronto ma vuoto.**

3 **Per creare una nuova tabella click su *Tables*, dalla toolbar selezionare *New Table*, apparirà la schermata che consente di inserire i dati nella tabella.**

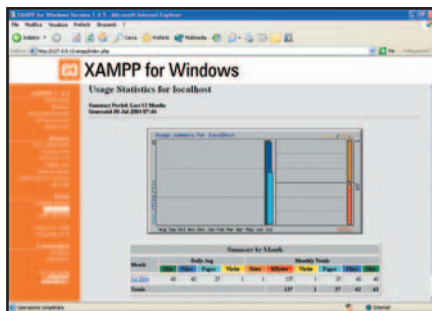
SOURCE FORGE

EasyPHP 1.7
Sistema completo per creare applicazioni web con PHP, MySQL e Apache

EasyPHP installa e configura automaticamente un ambiente di sviluppo completo per la creazione di applicazioni Web server side complesse, che interagiscono con i database, utilizzando il linguaggio di scripting PHP, il server web Apache e il database server MySQL. Inoltre, include il programma phpMyAdmin, un'applicazione Web based che semplifica notevolmente la gestione di database MySQL grazie ad un'interfaccia semplice ed intuitiva. EasyPHP è disponibile per sistemi operativi GNU/Linux (piattaforma LAMP – Linux Apache MySQL PHP) e MS Windows (piattaforma WAMP – Windows Apache MySQL PHP). La versione 1.7 include i seguenti software: Apache 1.3.27, PHP 4.3.3, MySQL 4.0.15 e phpMyAdmin 2.5.3.

easyphp1-7_setup.exe

XAMPP for Windows 1.4.5
ServerWeb, FTP, DBMS, PHP e Perl tutto in uno

Molti sviluppatori sanno, per esperienza personale, quanto sia difficile installare il server Web Apache e successivamente aggiungere e configurare il supporto per PHP, MySQL, Perl o altro. L'obiettivo degli sviluppatori di XAMPP è quello di fornire agli sviluppatori un modo semplice e veloce per installare una piattaforma di sviluppo Web completa, che ruota attorno al server Web Apache (capostipite di questa tipologia di software è stato EasyPHP). Al momento sono disponibili le versioni per GNU/Linux, Solaris e MS Windows. Quest'ultima, disponibile per i sistemi operativi Windows 98, NT, 2000 e XP, installa in un colpo solo Apache, MySQL, PHP più PEAR, Perl, mod_php, mod_perl, mod_ssl, OpenSSL, phpMyAdmin, Webalizer, Mercury Mail Transport System e NetWare Systems 3.32, JpGraph, FileZilla FTP Server, mcrypt, Turck MMCache, SQLite, e WEB-DAV + mod_auth_mysql.

Xampp-win32-1.4.5.zip

PHP-Nuke 7.3
Sistema per la creazione e la gestione di portali Web

Content Management System (CMS) o Web Portal System, potente e professionale realizzato interamente in PHP. Il software necessita del server Web Apache e di un database server SQL come MySQL, mSQL, PostgreSQL, ODBC, ODBC_Adabas, Sybase o Interbase (quasi sempre si tratta di MySQL). L'obiettivo di PHP-Nuke è consentire la creazione di siti Web automatizzati, principalmente per la gestione e pubblicazione di articoli e news con un potente sistema per la gestione degli utenti. Tra le caratteristiche principali spiccano una sofisticata GUI basata su Web per l'amministrazione del sistema, statistiche, gestione temi, banner, un potente motore di ricerca, un forum e il supporto per 25 diverse lingue, tra cui l'italiano.

PHP-Nuke-7.3.zip

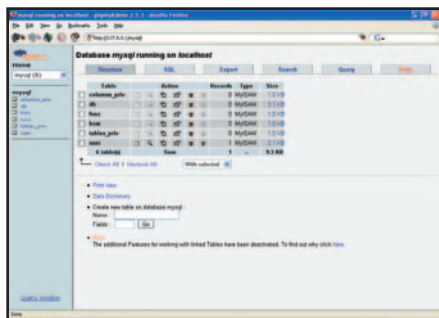
PostNuke 0.726
Una valida alternativa a PHP-Nuke

Il termine tecnico per indicare la categoria di appartenenza di PostNuke è CMS (Content Management System). Il sistema è molto simile a PHP-Nuke (pioniere nel campo dei CMS) e viene utilizzato per la gestione di siti Web complessi e professionali, i cosiddetti portali. Il software come il suo predecessore è interamente realizzato in PHP e utilizza MySQL come database server. Infatti, questi software basano la gestione dei contenuti, operata in modo dinamico, esclusivamente sull'utilizzo di database. La gestione del portale avviene totalmente mediante un pannello di amministrazione direttamente online, raramente è necessario ricorrere a client FTP, operazione necessaria solo nei casi in cui si decide di installare qualche nuova caratteristica. La gestione dei contenuti può essere effettuata da un qualsiasi PC connesso ad Internet, dall'amministratore del sito o direttamente dagli utenti precedentemente registrati, sempre utilizzando la stessa interfaccia di amministrazione, ovviamente con minori possibilità di intervento.

PostNuke-0.726-3.zip

phpMyAdmin 2.5.7
Interfaccia per l'amministrazione di server MySQL

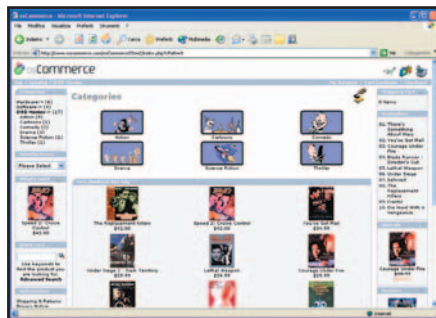
phpMyAdmin non è altro che un'interfaccia grafica (GUI - Graphical User Interface) che consente di amministrare database contenuti all'interno di un server MySQL. In pratica, permette di visualizzare il contenuto del database, creare, modificare, cancellare intere tabelle o singoli record; fare un backup dei dati e visualizzare altri tipi di informazioni che riguardano il database. Il programma è realizzato interamente in linguaggio PHP, quindi tutte le operazioni di amministrazione avvengono completamente mediante un comune browser Web. phpMyAdmin consente di gestire tutti gli aspetti del server MySQL e

non solo i dati contenuti all'interno del database. Infatti, permette di creare e cancellare interi database, gestire gli utenti e i permessi, riavviare il server, esportare i dati in formati diversi (CSV, XML e Latex) e creare documenti PDF. Il programma è disponibile in 44 differenti lingue.

phpMyAdmin-2.5.7-pl1.zip

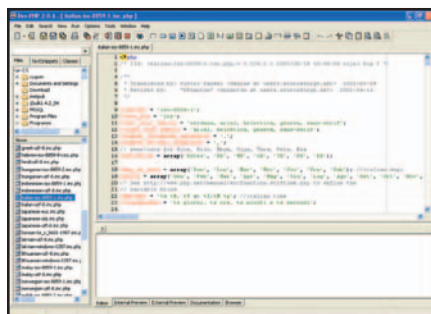
XOOPS 2.0.7
Il CMS alternativo

XOOPS, abbreviazione di eXstensible Object Oriented Portal System, è a tutti gli effetti un CMS, quindi, si tratta di un sistema automatizzato per la gestione di articoli, news e contenuti vari pubblicati sul Web, lo strumento ideale per sviluppare commmunity, portali aziendali, weblogs e molto altro. Il software è destinato principalmente alla gestione di portali Web in una Intranet oppure su Internet. L'amministratore controlla totalmente la gestione del sito attraverso un intuitivo e funzionale pannello di controllo, che consente di gestire gli utenti, inserire notizie, nuove pagine, download, link, faq e tanto altro ancora. Il sistema è sviluppato da un team internazionale di programmatori (*www.xoops.org*) che utilizzano il linguaggio PHP (molto diffuso per questo tipo di applicazioni) e si appoggia a un database MySQL, così come la maggior parte degli CMS. I siti creati con XOOPS sono interamente aggiornabili e gestibili via Web. E' disponibile anche il sito italiano ufficiale dedicato a XOOPS all'indirizzo *www.xoopsit.net*.
xoops_2.0.7.zip

osCommerce 2.2
Piattaforma Open Source per il commercio elettronico

Applicazione distribuita con licenza GNU GPL per creare e gestire un sito di e-commerce con il minimo sforzo e senza sostenere alcuna spesa. La piattaforma è realizzata interamente in PHP e la gestione dei dati riguardanti i prodotti e le diverse attività è affidata al server di database MySQL. osCommerce può essere eseguito su qualsiasi server Web che supporti PHP3 e PHP4, mentre per quanto riguarda la piattaforma hardware è perfettamente compatibile con i sistemi operativi GNU/ Linux, BSD, Solaris e Microsoft Windos. L'interfaccia utente è multi lingua (disponibile anche in italiano), personalizzabile graficamente, mentre il processo di installazione è molto semplice e avviene mediante un comune browser Web.

oscommerce_2.2.zip

Dev-PHP 2.0.6
IDE per lo sviluppo di script PHP

Ambiente di sviluppo integrato per la creazione di script e applicazioni mediante il diffuso linguaggio di scripting PHP e la libreria PHP-GTK (inclusa nel pacchetto PHP). Il software è Open Source rilasciato con licenza GNU GPL. Dev-PHP dispone di una serie di strumenti, tipici degli ambienti di sviluppo integrati, che semplificano e velocizzano enormemente lo sviluppo delle applicazioni: syntax highlighting, debugger PHP integrato, scorciatoie personalizzabili, numerose funzioni e parole chiave preimpostate e tanto altro.
Il programma è stato creato utilizzando l'ambiente di sviluppo Delphi/Kilyx ed è perfettamente compatibile con tutte le versioni dei sistemi operativi Microsoft, da Winodws 95 a XP.

devphp206.zip

PHPEclipse
Plugin PHP per l'IDE Eclipse

Il progetto PHPEclipse mira a creare un ambiente di sviluppo integrato per il linguaggio di scripting PHP. Per raggiungere tale scopo, gli sviluppatori non sono partiti da zero, ma hanno preferito sviluppare un "semplice" plugin che integri il supporto PHP nel potente IDE Eclipse. Quest'ultimo è un ambiente di sviluppo integrato multi linguaggio e multi piattaforma, creato appositamente con una struttura estendibile mediante l'utilizzo di plugin. Praticamente, così facendo, è possibile estendere le funzionalità per un numero pressoché illimitato di linguaggi di programmazione. Per quanto riguarda PHP, questo sistema ha portato, mediante PHPEclipse, alla realizzazione di un completo ambiente di sviluppo dotato di parser PHP, debugger, un sistema di template e formattazione del codice, e il supporto per database server SQL come MySQL e PostgreSQL.
phpeclipse_072004.zip

Apache2Triad 2.2
Piattaforma di sviluppo totalmente basata su Apache

Apache2Triad rappresenta l'estremizzazione del concetto che sta alla base di prodotti come EasyPHP, XAMPP, WAMPP e LAMPP, infatti, il pacchetto installa in un colpo solo e senza complesse configurazioni aggiuntive, un completo ambiente di sviluppo per numerosi linguaggi di programmazione (ASP, PHP, Perl, Python,Tcl) e diversi software di rete (FTP, HTTP, SSL, e-mail), compresi programmi per la gestione della sicurezza. Tra i software disponibili in bundle con il pacchetto sono presenti: Apache 2.0.47, MySQL 4.0.13, OpenSSL 0.9.7b, PHP 4.3.3, Perl 5.8.0, Tcl 8.4.2, Python 2.2.2, PHP-Nuke 6.8, PHPmyadmin 2.5.2, AWStats 5.7, Stunnel 4.04, SSLCert 1.0, Xmail 1.1.6 and SlimFTPd 3.13, UebiMiau 2.7.2 e PHPXmail 0.33. Il software è Open Source con licenza MIT.

apache2triad1.2.2.exe

Java 2 SDK, Standard Edition 1.4.2_05

L'indispensabile ambiente di sviluppo SUN

In attesa della versione definitiva di Tiger (nome in codice del JDK 1.5), quella che presentiamo resta la scelta obbligata per tutti coloro i quali vogliano sviluppare applicazioni Java. In questa versione, nuove funzionalità e migliori prestazioni arricchiscono l'ambiente. Rich client application e Web Services sono i campi in cui sono più evidenti i miglioramenti. Tra gli aggiornamenti che più faranno gola agli sviluppatori ci sono sicuramente quelli inerenti Swing. Davvero ghiotti i due nuovi look-&feel: GTK+ e, finalmente, Windows XP.
Anche le applicazioni Java possono così integrarsi pienamente nell'ambiente visuale del più recente Windows. Anche GTK+ risulta molto interessante: attraverso un semplice resource file, è possibile settare i parametri fondamentali del look &feel. Indispensabile al fine di utilizzare correttamente Eclipse.
j2sdk-1_4_2_05-windows-i586-p.exe

DBAny 1.0

Gestisci qualsiasi database

Un unico client per più database: Oracle, Sybase, DB2, MySQL, MSAccess, sono solo alcuni dei DB supportati da questa comoda applicazione. È possibile compiere qualsiasi operazione di aggiornamento o interrogazione, facilitando il travaso di informazioni da un DB all'altro. L'interfaccia è particolarmente efficace in caso di data entry "massiccio": la funzione Auto-Insert consente di evitare la re-immissione di informazioni ripetute.
Versione di prova, alcune funzioni risultano limitate.
DBAny.msi

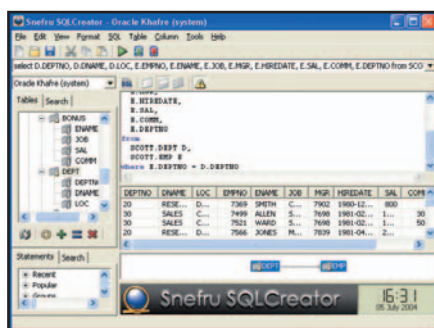
Relsoft DLL Compiler 1.0

Costruisci le DLL da Visual Basic

Un plug-in per Visual Basic che permette di creare DLL C-Style, utilizzando direttamente il noto ambiente RAD di Microsoft. Le DLL così compilate possono essere richiamate da qualsiasi applicazione C, C++, ASM o Delphi. Le opzioni disponibili consentono numerose ottimizzazioni in base alla dimensione della libreria e alla velocità di esecuzione. Versione dimostrativa, è possibile effettuare un massimo di dieci compilazioni.
DllCompilerDemo.zip

SQLCreator 1.0

Costruire query e interagire con qualsiasi RDBMS

Un applicazione completamente gratuita che consente di scrivere query SQL per qualsiasi RDBMS, inclusi: Oracle, DB2, SQLServer, MySQL, PostgreSQL e ODBC. Tra le funzionalità salienti si annoverano la gestione di connessioni multiple, l'evidenziazione sintattica ed il completamento automatico del codice. Molto utile risulta la visione ad albero disponibile per ogni connessione che pone in evidenza tutti gli oggetti disponibili per il relativo DB. E' possibile ottenere una rappresentazione grafica delle relazioni esistenti fra le tabelle e, su questa base, costruire le relative query.

SQLCreator 1.0.0.exe

Runprog 1.0.28

Aggiornare automaticamente il front end delle applicazioni Access

Funzionale e gratuita, questa utility consente di aggiornare in modo del tutto trasparente il front end delle applicazioni Access. Ogni volta pubblicherete un nuova versione del fornt end, tutte le macchine client saranno aggiornate all'avvio dell'applicazione. Semplice e gratuito.
RunProg.zip

IncUpdate 1.1

Aggiorna le tue applicazioni via Web

Dieci minuti per cambiare le tue applicazioni in modo che si aggiornino via Web: è quanto promette questo interessante tool. Risulta indipendente dal linguaggio utilizzato per lo sviluppo del software. Versione di valutazione valida quindici giorni.

incupdate_setup.exe

XEUS (XML for Easy User interface Scripting) 1.0

Testa le tue applicazioni in modo automatico

Un tool che non dovrebbe mancare nella cassetta degli attrezzi di ogni sviluppatore: XEUS si occupa di registrare tutte le azioni compiute da un utente, registrandole in uno script XML. Lo script può essere quindi modificato ed eseguito più volte al fine di testare il comportamento delle applicazioni. Richiede che sia installato il runtime di Java 1.4.02_03 o superiore. Versione dimostrativa, è possibile registrare un massimo di trenta eventi per ogni script.
XEUS1_0Setup.exe

MySQL Turbo Manager Enterprise 3.1

Gestisci gli oggetti del tuo DB MySQL

Un tool per amministratori e sviluppatori che abbiano adottato la piattaforma MySQL: è possibile gestire i privilegi, costruire query SQL, estrarre informazioni, esportare e modificare i dati presenti nel DB. Interessanti le funzioni di project management che semplificano il lavoro di gruppo. Versione di prova valida quindici giorni.
installer_mysql_turbo_manager_enterprise.exe

DarkBASIC Professional 1.0

La libertà di creare giochi 3D semplicemente

Un ottimo compilatore che consente di utilizzare il BASIC per creare giochi e ambienti virtuali tridimensionali. Il motore tridimensionale è stato creato sulla base di Quake 3D ed è possibile infatti utilizzare tutti i livelli e gli effetti di Quake. Versione dimostrativa valida trenta giorni.
darkbasic_professional_trial_v53.zip

TestTrack Pro 6.1

Tracciare bug e richieste degli utenti

TestTrack è un tool cross-platform per la catalogazione di bug e la pianificazione dello sviluppo software. Può essere inter-

rogato via browser e si integra perfettamente con Visual Studio 6, Visual Studio .NET e Visual SourceSafe. Permette agli utenti Windows e Macintosh di accedere simultaneamente allo stesso database cross-platform. Inoltre, include ottime funzioni per il filtering, permettendo di creare delle liste di problemi organizzate per priorità. Ottima la possibilità di generare dei report. La notifica via mail di tutti gli aggiornamenti al DB ed il pieno supporto a XML rendono sicuramente interessante questo prodotto. Licenza di valutazione della durata di trenta giorni, è possibile tracciare un massimo di 15 bug.
ttprowininstalldl.exe

Xassist 1.2

Un correttore di bozze per XML

Verificare e correggere la struttura di documenti XML e SGML: è quello che garantisce questo interessante tool. L'interfaccia è semplificata da un largo uso del drag&-drop ed il parser per i file XML può essere utilizzato con profitto nella fase di controllo della qualità.
Nella suite è inclusa la possibilità di analizzare gli schemi DTD. Versione di prova valida trenta giorni.
xassist.zip

Database Architect 1.0

Un piccolo aiuto per la progettazione dei DB

L'interfaccia semplice ed efficace di questo piccolo tool consente di specificare graficamente le proprietà di un Database relazionale. Purtroppo non è disponibile alcuna funzionalità per importare schemi da DB esistenti. In compenso, in base allo schema realizzato, Database Architect si occupa di generare gli script SQL necessari alla creazione del relativo database.
Versione dimostrativa.
da.zip

Universal SQL Query Tool 1.12

Gestire qualsiasi DB

Ancora un'applicazione per connettersi e gestire qualsiasi DB. Questa volta, leggerezza e semplicità sono le caratteristiche chiave. All'atto dell'installazione vi verrà chiesto un numero di serie: lasciando il campo vuoto, avremo la possibilità di provare l'applicazione per trenta giorni.
Install.exe

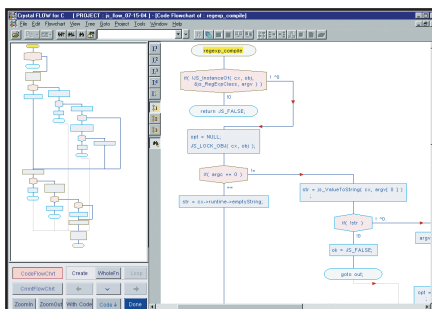
ActiveX Registration Manager 3.7.7

Gestire gli ActiveX

Tutti i programmatori, e non solo loro, devono combattere quotidianamente con la selva di ActiveX che affolla Windows. Questa applicazione promette di risolvere per sempre questo incubo: la funzione più importante è quella che permette di installare più versioni dello stesso ActiveX, consentendo di switchare in qualsiasi momento fra le versioni disponibili.
Licenza di prova valida quattordici giorni.
setup.exe

Crystal Flow for C 1.1

Per "familiarizzare" con il codice C

Una indispensabile utility per gli sviluppatori che debbano affrontare codice legacy scritto in C. Spesso, anche se il codice l'abbiamo scritto noi, capire il funzionamenti di un'applicazione può essere veramente arduo.
Crystal Flow si occupa di generare il flow chart relativo alla porzione di codice che indichiamo, velocizzando notevolmente il tempo necessario a comprenderne la struttura. Versione dimostrativa.

CrystalCFLOW_EvalADv1.zip

TextPad 4.7.3

Un editor testuale completissimo

Chi è stufo del solo text pad di Windows, potrà trovare in TextPad pane per i suoi denti. TextPad si presenta come uno dei migliori editor di testo per Windows.
Programmando, siamo sempre alla ricerca di un buon editor che sia al contempo semplice e veloce e che non faccia rimpiangere le funzionalità dei word processor più avanzati. In TextPad si segnala addirittura la presenza di un registratore di macro, utilissimo a velocizzare le operazioni più frequenti.
Utilissima è anche la funzione di split che consente di avere più viste sullo stesso documento.
txpeng473.exe

Install-Us Professional 4.503

Per creare installazioni per Windows di livello professionale

I pacchetti di installazione che si possono creare con Install-us hanno un aspetto decisamente professionale e possono essere utilizzati per distribuire il nostro software su qualsiasi media: CD, Internet o semplici dischetti. E' presente una completa gestione della funzione di disinstallazione. In sei passi, è possibile specificare tutti i dettagli relativi alla corretta installazione dell'applicazione che vogliamo distribuire. Versione dimostrativa.
iuse.zip

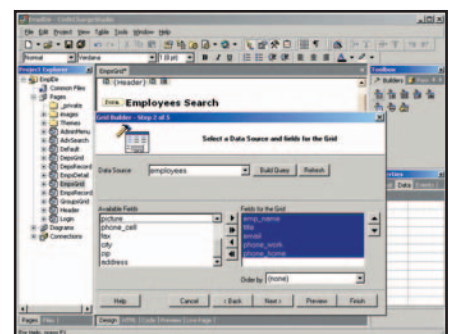
Excelsior JET 3.6

Converte applicazioni Java in Win32

Il tool che vi presentiamo contiene al suo interno una Java Virtual Machine che, attraverso delle particolari ottimizzazioni, permette di nascondere agli utenti la natura "Java" dell'applicazione. I componenti della Virtual Machine sono incapsulati in librerie DLL e consentono di distribuire in modo più semplice il nostro software. Versione di prova valida sessanta giorni.
jet-latest-eval-win32.exe

CodeCharge Studio 2.3

Creare applicazioni senza scrivere codice

La soluzione proposta da CodeCharge Studio consente di sviluppare applicazioni visualmente, creando Web Application database-driven senza scrivere una riga di codice… o quasi. Rivolto a sviluppatori di qualsiasi esperienza, consente di risparmiare moltissimo tempo in produzione e permette di trasformare qualsiasi database (MS Access, MS SQL, MySQL, Oracle) in un'applicazione Web perfettamente funzionante. È incluso un plug-in per FrontPage.
Versione di prova valida venti giorni.

CCStudio2_3.exe

Microsoft Tech·Ed 2004

L'edizione di quest'anno è stata incentrata più sul consolidamento delle tecnologie esistenti che sulla presentazione di nuove piattaforme. È stata anche l'occasione per fare il punto sulla sicurezza dei sistemi secondo Microsoft

dal nostro inviato Raffaele del Monaco

Amsterdam, il paese dei tulipani e della tolleranza. Quale posto migliore per festeggiare la versione europea della pax informatica secondo Microsoft: finita l'epoca delle guerre di religione e dei tabù (Java e Linux su tutti), è il momento del dialogo e dell'interoperabilità. Gli infiniti canali che caratterizzano il paesaggio urbano di Amsterdam sembrano una perfetta metafora per descrivere la nuova visione della programmazione e dei sistemi targati Microsoft: la ricchezza viene dall'apertura e dalla possibilità di interagire con gli altri.
Come per ogni TechEd, la città ospite dovrà subire l'invasione di migliaia di soldati dello sviluppo, tutti con l'uniforme d'ordinanza: borsa-ricordo e occhiaie comprese. Queste ultime soprattutto saranno ricordate in una città in cui è davvero difficile andare a letto presto!

Fig.1: *Amsterdam: la migliore cornice possibile*

OCCHI CHIUSI, OCCHI APERTI

Entrando nella enorme sala che ospiterà la keynote, veniamo accolti da un frastuono incredibile: sul palco, un gruppo di musicisti africani suona con gran vigore tamburi di varie fogge. Sarà il leit motiv di tutto il tech-ed. Cercando un posto libero in sala, scopriamo che su ogni sedia è presente un piccolo bongo: un gadget davvero sui generis che diventa ancora più gradito quando si scopre che viene direttamente dall'Africa e che ognuno è stato acquistato direttamente sul posto, pagando le persone che li hanno costruiti. La volontà di mostrare il lato umano dell'azienda non termina qui. Tra lo stupore di tutti, il primo a salire sul palco è George Cathomas, uno sviluppatore svizzero non vedente. Aiutato da una tastiera braille e dal feedback vocale delle applicazioni, Cathomas dimostra quando siano efficaci gli attuali strumenti a disposizioni dei non vedenti. L'intervento si chiude con un sacrosanto appello a noi sviluppatori: cerchiamo di fare applicazioni e siti che tengano conto delle esigenze di chi ha avuto meno fortuna di noi.

LE NOVITÀ

È stato dunque il turno di Jonathan Murray, Vice Presidente di Microsoft e vero padrone di casa. Molti dei presenti, compreso il sottoscritto, erano reduci dalla scoppiettante PDC di Los Angeles, in cui annunci e sorprese avevano tenuto desta l'attenzione per tutta la durata dell'evento. Il TechEd di quest'anno è invece stato alquanto parco di novità: quella più importante è senza dubbio l'annuncio di una nuova famiglia di prodotti di sviluppo battezzata "Express" caratterizzata da un prezzo molto basso e da funzionalità ridotte rispetto a Visual Studio. Questo l'elenco completo dei nuovi strumenti: Visual Basic 2005 Express, Visual C# 2005 Express, Visual C++ 2005 Express, Visual J# 2005 Express, Visual Web Develo-

Fig. 1: *La keynote si è aperta con oltre 6000 persone a suonare i bonghi contemporaneamente: è stata chiesta la certificazione al Guinness dei primati. Sul serio!*

per 2005 Express, Sql Server 2005 Express. La disponibilità si avrà a partire dal 2005, mentre i prezzi ancora non sono noti. A parte Sql Server 2005 Express, che sarà gratuito, gli altri prodotti dovrebbero tutti posizionarsi sotto i cento dollari.
Non è stato ben chiarito in cosa si differenzieranno le versioni Express da quelle standard, pare che le principali differenze risiedano nel mancato supporto per gli Add-in, l'impossibilità di effettuare il debugging in remoto, la mancanza del supporto per lo sviluppo mobile, e altre differenze minori che crediamo non fermeranno i potenziali acquirenti visto il basso prezzo prospettato. La sensazione è che, per quanto importante, questo annuncio sia stato accolto con una certa freddezza dalla platea, composta per lo più da professionisti dello sviluppo e dipendenti di aziende medio-grandi, tutti soggetti cui non è certamente rivolta la famiglia Express. Eppure le speranze di Microsoft sono grandi: dopo aver raggiunto livelli di eccellenza nella fascia alta dello sviluppo con .Net, l'immagine del gigante di Redmond nel campo della programmazione Entry-level sembra decisamente appannata. Mentre VB 6 è in via di pensionamento, le nuove proposte che si sono succedute negli ultimi anni non sono mai riuscite a eguagliarne semplicità ed efficacia. D'altronde, i milioni di sviluppatori su cui ora può contare Microsoft sono in buona parte dovuti all'effetto volano di VB 6: guai a perdere il traino dei prodotti rivolti a chi si avvicina alla programmazione!
Gli altri annunci della keynote hanno riguardato il nuovo Team System che sarà parte integrante di Visual Studio 2005 e la forte integrazione che sarà possibile proprio fra VS 2005 ed il Netweaver di SAP.

SEMPLICITÀ NELLO SPAZIO

Quest'anno la Mobile Developer Conference americana non ha avuto il suo corrispondente europeo: tagli di budget o altre imperscrutabili motivazioni hanno indotto Microsoft a incorporare nel TechEd la conferenza dedicata allo sviluppo Mobile. Nella keynote, due speaker travestiti rispettivamente da PocketPC e da SmartPhone hanno messo su una bella demo live: con le solite "poche righe di codice" hanno sviluppato un client per dispositivi mobili in grado di catturare immagini attraverso la fotocamera integrata, inviandole poi ad un blog.
Molto più efficace e impressionante, una demo su smartphone: grazie ad un accordo con gli operatori telefonici era possibile tracciare la posizione dei telefonini e, incrociando i dati con le cartine offerte da Mappoint, si potevano seguire in tempo reale gli spostamenti degli utenti che "accettavano" di essere sorvegliati. I dati forniti in merito sono stati abbastanza scarsi: probabilmente il metodo utilizzato è la triangolazione possibile con i ripetitori GSM. In pratica, lo stesso metodo adottato dalla polizia, previa autorizzazione del magistrato. Ed ecco il punto: dove va a finire la privacy? Si dirà che è necessario che l'utente accetti di essere localizzato ma, ad esempio, un dipendente di un'azienda sarà realmente libero di rifiutare un simile sistema di controllo.
Inutile star qui a sottolineare che non è compito di un giornalista specializzato approfondire questioni di natura etica, eppure credo sia arrivato il momento di cominciare a scegliere le innovazioni realmente utili, imparando a rifiutare quelle che reputiamo dannose. Ovviamente, i vantaggi nei settori della sicurezza e dell'assistenza sanitaria potrebbero essere numerosi e significativi. Ci sembra comunque il caso di sottolineare i pericoli legati agli abusi di una tecnologia come questa: qualche anno fa si discuteva della costituzionalità di un braccialetto per detenuti che ne consentisse la localizzazione nei momenti di libertà vigilata. Ma è questo l'incanto della società contemporanea: non imporre nulla, lasciando che ognuno vada a comprare il proprio braccialetto, in cambio dell'ennesima fettina di libertà.
A chiudere la Keynote è arrivato l'affascinante intervento di Jim Gray che ha illustrato il progetto SKYSERVER, che permetterà agli studiosi di astronomia di tutto il mondo di interrogare una enorme base di dati distribuita. Quello che si realizzerà sarà una sorta di "World Wide Telescope", sorta di telescopio virtuale che, raccogliendo l'enorme massa di dati già disponibili in Internet, li metterà a disposizione in una forma coerente e interrogabile via Web.

Fig. 4: *Jim Gray, la star più acclamata del TechEd, ha presentato il titanico progetto Terraserver*

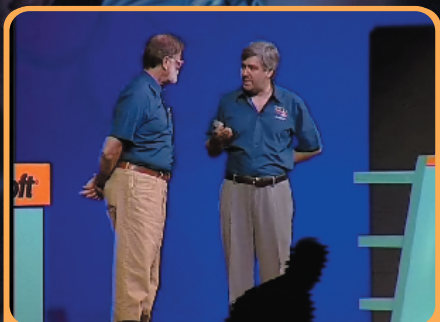



Fig. 4: *Jonathan Murray presenta Jim Gray*

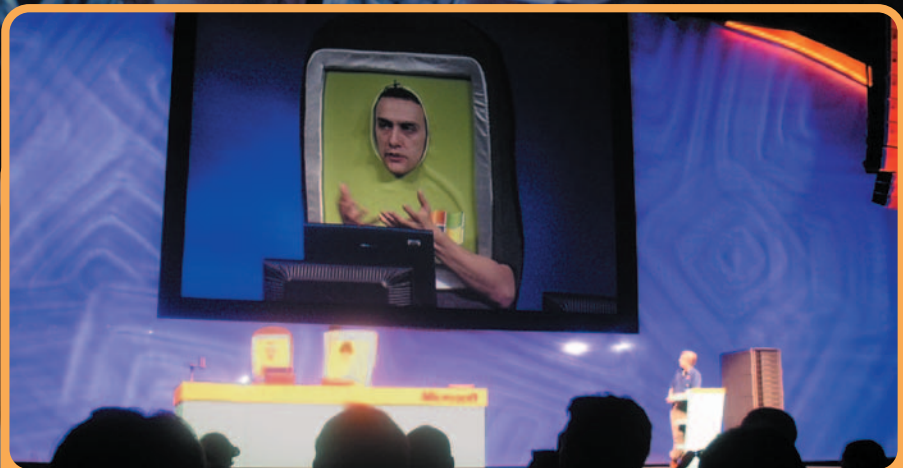



Fig. 4: *Jonathan Murray assiste divertito alla demo Mobile*

FULL IMERSION

La settimana del TechEd è scivolata, come sempre, fra centinaia di sessioni in parallelo e corse a rompicollo per raggiungere le sale dedicate agli speech. Il pubblico, composto per lo più da professionisti dell'informatica, sembrava particolarmente interessato alle problematiche riguardanti la sicurezza, riguardo alle quali, trovate più avanti il resoconto di una lunga intervista con uno dei maggiori responsabili Microsoft.
Tra le sessioni più affollate, si sono segnalate quelle dedicate al rapporto con l'Open Source: sappiamo tutti che questo movimenti è in buona parte avversato da Microsoft, tuttavia Bill Gates sembra aver preso coscienza che il futuro vedrà l'Open Source come attore fondamentale con cui il dialogo potrebbe dare molti più frutti di una guerra.
Altro argomento molto dibattuto è stato il Voice-Over-IP, soprattutto perché mamma Microsoft aveva messo a disposizione centinaia di PC con un collegamento VOIP attivo, attraverso cui era possibile chiamare in tutto il mondo… gratuitamente! Non vi dico l'affollamento: sembrava che molti dei presenti fossero volati ad Amsterdam al solo scopo di poter chiamare a casa.
Anche questa volta, chi ha partecipato tornerà a casa non solo con qualche nozione in più ma anche con una migliore conoscenza del mondo della programmazione in Europa: più business e meno sogni, questa la direzione che sembra aver preso l'informatica vista da qui.

INTERVISTA A ANDREA SILVESTRINI

Come da molti anni a questa parte, il tema della sicurezza era al centro dei discorsi di tutti ed era il vero motore di molti speech. Abbiamo affrontato l'argomento con Andrea

Fig. 4: *Don Box, icona vivente della programmazione Service Oriented*

Fig. 4: *L'intrepida via della programmazione Office via .Net*

Silvestrini, Marketing Director della Security Initiative.

Andrea Silvestrini: Qualcuno potrebbe pensare "oddio l'ennesima iniziativa della Microsoft" dopo il Trustworthy Computing ora se ne escono con qualcos'altro. In realtà Security Mobilisation Initiative e trustworthy computing sono complementari: abbiamo lanciato il Trustworthy Computing due anni e mezzo fa, nel gennaio 2002. A quel tempo Bill Gates si era reso conto che serviva dare una scossa al gruppo sviluppo prodotti, circa 23-25.000 persone a Redmond. Sostanzialmente si trattò di mettere la priorità sulla sicurezza rispetto alle altre cose, quindi Trustworthy Computing nasce come modo di svegliare tutto il gruppo sviluppo prodotti e definire quelle che erano le priorità dell'azienda: sicurezza prima di tutto, features e tutto il resto venivano dopo.

ioProgrammo: In che modo e in che misura Microsoft è coinvolta in questa iniziativa?
AS: Trustworthy Computing ha una fortissima componente tecnologica. L'impatto che ha avuto sul reparto vendite e marketing è stato molto limitato: un anno fa, se chiedevi a qualcuno della Microsoft Italia qualcosa sulla sicurezza, lui pensava che fosse una responsabilità interamente del gruppo prodotti. Diceva "si, abbiamo un po' di problemini… la colpa è loro."
Invece, la problematica della sicurezza è molto più complessa e non può essere risolta solo da un approccio puramente tecnologico. Quello che costituisce l'ambito della sicurezza in ogni azienda è sostanzialmente il risultato di tre vettori principali: la tecnologia, le persone e il processo.
Con il Trustworthy Computing abbiamo indirizzato la componente di tecnologia, introducendo una serie di check sulla qualità, sulla review del codice etc. Ma i prodotti da soli non bastano. Molto dipende da come sono utilizzati: dai processi e dalle policy adottati. Per cui quest'anno, soprattutto dopo Blaster

Fig. 4: *L'infermiera di Zantaz proponeva una cura per le difficoltà di organizzazione e integrazione di mail e database www.zantaz.com*

Andrea Silvestrini Marketing Director, Security Initiative per Microsoft EMEA

che è stato un po' la sveglia per tutti, ci siamo resi conto che se non lavoravamo sulle persone e sui processi allo stesso momento, non avremmo avuto successo. Lì nasce la Security Mobilization Iniziative. In estrema sintesi, uno può vedere la Trustworthy Computing come qualcosa che riguarda il gruppo prodotto, mentre la Security Mobilization Iniziative è qualcosa che riguarda le vendite e il marketing: tutte quelle persone e quegli impiegati che lavorano con i clienti e con i partner. Serviva e serve un grosso sforzo di

Fig. 4: *Mainsoft presentava un ambiente capace di far girare il bytecode di Java direttamente su .NET www.mainsoft.com*

Fig. 4: *Nel pittoresco stand di Captaris era possibile provare degli interessanti sistemi di integrazione www.captaris.com*

education and awarness, quindi alzare la sensibilità di tutti sulle problematiche della sicurezza.

ioP: Da più parti ho sentito diverse lamentele sul ciclo di rilascio delle patch: chi ne denunciava il ritardo, chi l'eccesso. C'è intenzione di cambiare qualcosa?

AS: Noi in genere facciamo quello che i clienti chiedono: quando approntiamo delle strategie, andiamo a chiedere ai clienti e ai partner che cosa vogliono. Prima la strategia era: sorgeva un problema, rilasciavamo una patch. I network Administrator, o chi per loro, ci hanno detto forte e chiaro che erano stufi. Vivevano in perenne apprensione: magari ci si trova in ferie, esce la patch e si è costretti a tornare in fretta e furia. Ci hanno dunque invitato ad avere una certa cadenza.

Noi abbiamo scelto la cadenza mensile perché quella che meglio si adattava alle dinamiche di insorgenza dei problemi, ma la richiesta di periodicità è venuta proprio dai clienti. Per venire in contro ad eventuali urgenze, abbiamo comunque le cosiddette "Out of cycle patch", che sono soltanto per delle vulnerabilità critiche. Se c'è una vulnerabilità critica e noi vediamo che c'è "un po' di attività", allora non è che aspettiamo un mese lasciando i clienti allo scoperto.

ioP: Per non ripetere un altro caso Sasser, oltre alle patch, cosa avete in mente?

AS: Parte del ruolo della Security Mobilization Iniziative è preparare Microsoft ad avere un buon sistema di mobilitazione in caso di emergenza. Con Sasser, ci siamo mobilitati una settimana prima che il virus uscisse: il nostro gruppo che monitorizza le hacking communities aveva visto che c'era molta attività attorno a questo virus e c'era già il codice su Internet e diversi code-builder ci stavano già lavorando. Per cui, prima ancora che Sasser uscisse, abbiamo cominciato a mobilitarci internamente e presso i clienti, sollecitandoli ad installare la patch. Quindi, siamo particolarmente contenti del fatto che Sasser sia stato molto meno grave di Blaster. Dal punto di vista tecnologico non sono molto dissimili, e quindi il fatto che abbia avuto un impatto molto limitato è tutto sommato la prova che il tipo di mobilitazione che attuiamo ha funzionato. Il processo che

Fig. 3: Il duro mestiere degli sviluppatori ha bisogno un po' di relax

Fig. 3: Office Systems ha messo in palio il premio più ambito: un robot telecomandato

abbiamo sviluppato prevede una diramazione dell'allarme che dall'headquarter di Microsoft arriva fino ai responsabili della sicurezza delle aziende nostre clienti. Da questo punto di vista posso affermare che sono stati fatti parecchi passi avanti.

ioP: Eppure, tra molti amministratori è diffusa l'idea che Linux sia di per se più sicuro. I tanti, troppi attacchi che i sistemi operativi Microsoft hanno subito, hanno generato una crescente irrequietezza in molti utenti: cosa ne pensa?

AS: A chi mi solleva questa obiezione, in genere dico: andiamo a guardare i dati e confrontiamo quante vulnerabilità hanno avuto i due sistemi in un dato intervallo temporale. Forrest ha di recente fatto una grossa rivalutazione di Windows Server rispetto ad una piattaforma Open Source. Quello che direi è che c'è tanto mito ed una certa animosità nelle discussioni, ma devo dire che, in particolare nell'Enterprise, le opinioni stanno rapidamente cambiando. Devo dire soprattutto grazie a Windows Server 2003, non a caso il primo prodotto sviluppato sotto Trustworthy Computing . Nei primi trecento giorni di Windows Server 2003 ci sono state nove vulnerabilità, contro le 37-38 dei primi trecento giorni di Windows Server 2000: quindi una riduzione del 75%. Con Windows Server 2003, solo la fase di sviluppo ha beneficiato del Trustworthy Computing.

Al contrario, Longhorn sarà anche architettato sotto Trustworthy Computing .

L'avere un'architettura che rispetta il Trustworthy Computing ha delle immediate ricadute, anche pratiche: uno dei più grossi problemi delle patch è il dover riavviare il sistema. I nostri clienti si sono spesso lamentati di questo aspetto: rispetto a un anno fa, abbiamo già ridotto il problema del 30%.

Ma il problema è architetturale: con Longhorn ci sarà l'85% di reboot in meno rispetto all'anno scorso.

ioP: Restando nell'ambito della sicurezza, credo che tra i problemi più gravi ci sia il rischio mono-coltura. L'avere lo stesso sistema operativo su oltre il 90% dei PC del pianeta innalza di fatto la pericolosità di qualsiasi virus. Si può fare lo stesso discorso che, in agricoltura, si fa per le coltivazioni OGM: perdendo la biodiversità si rischia una sorta di apocalisse improvvisa. Il fatto che ogni pianta sia diversa dall'altra è a la migliore garanzia per la loro (e per la nostra) sopravvivenza.

AS: La soluzione sarebbe ritornare ad una "balcanizzazione" dei sistemi operativi per poter risolvere un problema come questo. Anche se ce ne fossero tre a coprire il 100% del mercato avremo virus per tutti e tre. E' chiaro che avendo Microsoft oltre il 90% del mercato è un buon bersaglio, e la produzione continua di virus si può associare proprio al suo successo di mercato. Del resto la balcanizzazione è proprio ciò che ha abbattuto le quote di mercato di Unix: c'erano diversi "flavour" di Unix ed era diventato troppo costoso produrre applicazioni che fossero compatibili con tutte le versioni. Il cosiddetto network-effect presente in tutto l'information technology corrisponde al desiderio degli utenti di avere cose che funzionino sempre.

ioP: Torniamo dunque a una questione di priorità: la sicurezza passa al secondo posto dopo i risultati di mercato. Non è un po' pericoloso, stante l'importanza vitale di Internet e dei PC nella vita dell'Occidente?

AS: La prima regola resta comunque quella di andare in contro alla volontà dei clienti, ed il successo di Microsoft sta proprio a dimostrare che gli utenti preferiscono la standardizzazione. Gli utenti non vogliono avere n versioni di Windows che non funzionano

Fig. 4: Lo spazio dedicato agli Hands On Lab era sempre affollato: tutti curiosi di provare gli strumenti Express!

fra loro. Il problema per noi non è di scegliere fra standardizzazione e sicurezza, ma avere tutte e due. La nostra sfida è avere un unico S.O. e lavorare sulla sicurezza affinché quel sistema operativo sia il più possibile al riparo dagli attacchi.

ioP: Il punto di vista di Microsoft è chiaro, tuttavia è diffusa la percezione che gli utenti di Windows risentano dei problemi di sicurezza in modo più significativo rispetto chi ha adottato Linux.

AS: È chiaro che si vorrebbero sempre avere risultati nel breve termine, ma vediamo lo scenario: noi abbiamo circa 100 milioni di sistemi installati, di cui solo il 20% fa riferimento all'ultima versione. Ben 80 milioni di macchine utilizzano i vecchi S.O.: Win 98, Win 2000, ecc. Quindi, per quanto si possa lavorare bene nel Trustworthy Computing , ci si confronta sempre e comunque con la base installata. Ed è da tenere presente che, in ambito consumer, ci vogliono dai cinque ai sette anni prima che un utente cambi PC. C'è quindi una certa inerzia nel mercato. Man mano che Windows XP si affermerà, vedremo che ci sarà una crescente fetta di utenti che sarà incommensurabilmente più sicura, ma fino a che non avviene l'upgrade, possiamo spendere miliardi e miliardi di dollari, ma non produrranno alcun effetto su quelli che lavorano con Windows 98.

Noi siamo in questo momento come era l'industria dell'auto negli anni settanta: non c'erano cinture di sicurezza, non c'era l'ABS, non c'era l'EPS. E nessuno si lamentava. Negli anni ottanta è cominciato a sorgere un movimento che richiedeva più sicurezza che, partendo dalle auto più costose ha portato l'ABS fino alla Panda. Noi ci stiamo avviando verso una maggiore consapevolezza, come negli anni ottanta. Noi vogliamo diventare la Volvo dell'informatica, facendo della sicurezza un punto di forza anche per il marketing.

ioP: Beh, viste le dimensioni di Microsoft, il rischio è che diventi come la General Motors, la Ford e Volkswagen messe assieme. Pensate che riuscirete a fronteggiare l'avanzata dell'Open Source?

AS: Noi siamo leader perché, lavorando in un azienda strutturata, riusciamo ad integrare il nostro software meglio di quanto possa avvenire in un qualsiasi ambiente open source, dove non c'è una organizzazione. Le due filosofie di sviluppo non scompariranno di certo ma, magari fra sei anni, cambierà la percezione rispetto ai due approcci.

ioP: Crede che la strategia che Microsoft adotta per contrastare Worm e Exploit sia sufficiente a garantire la sicurezza degli utenti? Non è possibile intervenire per bloccare il circolo vizioso Patch-Exploit?

AS: Ad aprile, ci sono stati oltre 103 milioni di utenti che hanno scaricato la patch nelle prime tre settimane del mese. Ed è da considerare che, nove mesi prima, erano appena 10 milioni gli utenti che aggiornavano regolarmente il sistema. Questa è la strada principale per garantire maggiore sicurezza agli utenti, per cui continueremo a sviluppare l'update di Windows: un sistema con le patch aggiornate non ha problemi. Se si va a vedere la storia degli exploit non ne è uscito nessuno prima che noi approntassimo la patch.

Il processo è questo: qualcuno scopre la vulnerabilità, se è responsabile ne parla con noi altrimenti vanno sul Web e pubblicano un articolo divulgando la notizia. Noi abbiamo lavorato con i maggiori esperti di vulnerabilità ed ora molto pochi si possono definire irresponsabili, la maggior parte ne parla prima con noi: noi ci lavoriamo, le studiamo e rilasciamo la patch. Dopo nove giorni, qualcuno fa il reverse engineering della patch: la apre, la scompone e c'è qualcuno che mette sul Web come fare il virus.

Questo processo non si può interrompere, perché in America c'è il primo emendamento che protegge la libertà d'espressione: anche se tu metti su Internet come fare la bomba atomica, nessuno ti può dire niente. Quindi, se spieghi come costruire Blaster, non sei perseguibile penalmente. Probabilmente, almeno nelle persone normali, qualche dubbio dal punto di vista morale lo lasci, però non si può fare nulla. Quindi qualcuno, magari qualche ragazzino tedesco, prende queste istruzioni e costruisce il virus.

Per questo è fondamentale installare le patch appena le rilasciamo: perché non c'è virus prima. Paradossalmente, "loro" usano le nostre patch per creare il virus.

ioP: In conclusione, quanto inciderà il nuovo e chiacchieratissimo SP2 sulla sicurezza di Windows XP?

AS: Il SP2 sarà un grosso salto in avanti. Dopo aver analizzato tutti i virus, le vulnerabilità e gli exploit, abbiamo cercato di capire quali sono i maggior vettori di attacco e ne abbiamo identificati quattro: network attack, dovuto tipicamente a qualche porta lasciata aperta; malicious e-mail, con gli attachment che se cliccati lanciano un virus; malicious web surfing, con siti che lanciano pop-up e spyware vari; infine il buffer overrun. Se andiamo a vedere la storia degli ultimi sei anni, il 90% degli attacchi utilizzava uno di questi quattro vettori. Quindi non abbiamo lavorato sulla singola vulnerabilità, ma abbiamo chiuso le principali vie di attacco. Quindi, chi installerà il SP2 potrà beneficiare di un enorme miglioramento in termini di sicurezza, quantificabile in un ordine di grandezza rispetto agli attuali standard.

Fig. 4: Solito affollamento allo stand della Intel… la forza di un gadget! www.intel.com

Il codice e le tecniche per sfruttare le innovazioni di Tiger

Le novità di Java 1.5

Java 1.5 introduce numerose novità in termini di linguaggio vero e proprio. In questa nuova serie di articoli esamineremo, in modo dettagliato, le più importanti. Prepariamoci alla rivoluzione

Il nome in codice del progetto Java 1.5 è Tiger. Esso introduce numerose caratteristiche che pur non rappresentando delle novità assolute nel mondo della programmazione, cambieranno notevolmente il nostro modo di programmare in Java. Al momento, è possibile scaricare la J2SE 1.5 beta 1 dal sito della Sun, all'indirizzo: *http://java.sun.com/j2se/1.5.0/*. Tutti gli esempi proposti sono stati scritti e testati con questa versione.
Per compilare codice compatibile con la nuova sintassi di Java 1.5 è necessario specificare l'opzione *-source 1.5* del compilatore:

```
> javac -source 1.5 Prova.java
```

AUTOBOXING

In Java la distinzione tra tipo base e classe è piuttosto netta. In particolare, non è possibile assegnare ad un'istanza di una classe un valore appartenente ad un tipo base e viceversa. Ogni classe deriva implicitamente dalla classe *Object*; un tipo base invece non solo non deriva da *Object*, ma non può essere considerato neanche una classe. Infatti esso non ammette né campi né metodi. Sebbene esista questa netta distinzione, in alcuni casi è necessario che un valore appartenente ad un tipo base possa essere visto come istanza di una classe, per esempio quando si desidera assegnarlo ad un tipo *Object* generico. Ciò accade spesso quando si lavora con le collection. Questo problema viene risolto in Java con l'utilizzo delle classi *wrapper*. Il fine di queste particolari classi, è quello di "avvolgere" un valore (appartenente ad un tipo base) all'interno dell'istanza di una classe. Procedendo in questo modo, esso potrà essere visto come oggetto appartenente ad essa; inoltre, opportuni metodi restituiranno il valore originale come tipo base. Per esempio, la classe *wrapper* per il tipo base *int* è *Integer*. Essa possiede un attributo privato value di tipo *int* contenente il valore originale ed il metodo *intValue* che lo restituisce.
Il seguente esempio mostra come utilizzare la classe *wrapper Integer* per aggiungere un valore intero ad un *ArrayList*:

```
int i = 5;
ArrayList list = new ArrayList();
Integer wi = new Integer(i); //Wrapping
list.add(wi);
```

il valore intero potrà essere recuperato all'occorrenza nel seguente modo:

```
Integer wi = (Integer) list.get(0); //Unwrapping
int i = wi.intValue();
```

Esiste una classe *wrapper* per ogni tipo base. La tecnica di autoboxing è stata introdotta in Java 1.5 per avvicinare in un certo senso i tipi base alle classi. Questo però non significa né che Java 1.5 vede i tipi base come classi (come invece accade per .NET) né tanto meno che le classi wrapper siano diventate obsolete. L'autoboxing in pratica non fa altro che automatizzare il passaggio tra tipo base a classe *wrapper*. Con Java 1.5 è consentito assegnare un valore ad un'istanza della relativa classe wrapper utilizzando l'operatore =. Il codice che segue è un esempio di autoboxing per il tipo *int:*

```
Integer k = 5; //Autoboxing
```

La variabile *k* sarà quindi un riferimento ad un'istanza delle classe *Integer* dove il valore interno è 5.
È possibile assegnare direttamente un valore appartenente ad un tipo base anche ad una generica istanza di *Object,* come mostrato di seguito:

```
Object obj = 5;
```

Ciò che avviene a fronte di questa istruzione è la creazione implicita di un oggetto *Integer* che sarà riferito dall'istanza generica *obj*. In pratica è come scrivere:

```
Integer k = 5;
Object obj = k;
```

L'autoboxing può essere applicato a tutti i tipi base e alle relative classi *wrapper.* Nell'esempio che segue saranno utilizzate le classi *Boolean* e *Double:*

```
Boolean b = true;
Double d = 3.4;
```

Da questo punto in poi le istanze *b* e *d* sono oggetti veri e propri che contengono rispettivamente i valori *true* e 3.4.

AUTOBOXING E COLLECTION

Le classi *wrapper* si rivelano indispensabili quando si desidera assegnare un valore appartenente ad un tipo base ad un elemento di una collection. Supponiamo per esempio di voler creare un *ArrayList* di interi. Come sappiamo, l'*ArrayList* è una collection dinamica che ha come elementi oggetti generici. Essendo *int* un tipo e non una classe, non è possibile aggiungerlo ad un *ArrayList*. La soluzione al nostro problema sta nel creare un oggetto appartenente alla classe wrapper *Integer* contente il valore intero e di aggiungerlo all'*ArrayList*. Ecco il codice relativo a quanto detto:

```
// Tipi base e collection (Java 1.4)
ArrayList list = new ArrayList();
list.add(new Integer(10));
list.add(new Integer(5));
list.add(new Integer(3));
```

Con Java 1.5, utilizzando la tecnica di autoboxing, non sarà più necessario creare esplicitamente gli oggetti *Integer.* Ecco la nuova versione:

```
// Tipi base e collection (Java 1.5)
ArrayList list = new ArrayList();
list.add(10);
list.add(5);
list.add(3);
```

Come si può notare, il codice scritto per la versione 1.5 è più compatto e intuitivo.

UNBOXING

Il processo di *unboxing* (incredibile ma vero) è il processo inverso a quello di *autoboxing.* Come abbiamo visto, le classi *wrapper* e la tecnica di autoboxing sono utilizzate per trattare un valore al pari di un oggetto. Dopo le elaborazioni però, solitamente, si vuole ritornare al valore originale appartenete al tipo base. Prima dell'avvento di Java 1.5, tale operazione poteva essere effettuata utilizzando il metodo *xxx Value* della classe *wrapper* per il tipo *xxx*. Per il tipo *int,* ad esempio, si può utilizzare il metodo *intValue* della classe *Integer,* come mostrato dal seguente esempio:

```
// wrapping...
Integer k = new Integer(5);
//..unwrapping
int i = k.intValue();
```

Il valore intero 5 viene prima "wrappato" nell'oggetto *k* e successivamente assegnato all'intero *i* mediante il metodo *intValue*. Tale tecnica di *wrapping /unwrapping* in Java 1.5 si trasforma in *autoboxing /unboxing*. Ecco lo stesso esempio scritto per Java 1.5:

```
// boxing...
Integer k = 5;
//...unboxing
int i = k;
```

come si può notare l'operazione di *unboxing* può essere effettuata senza utilizzare la classe *wrapper Integer* ed il metodo *intValue.* Quando il valore è invece "wrappato" all'interno di un oggetto generico, l'*unboxing* non potrà essere automatico; è necessario specificare esplicitamente la classe *wrapper* opportuna:

```
Object obj = 8;
int i = (Integer) obj;
        //Unboxing da Object
```

Come si può notare, usando il *cast,* l'oggetto generico *obj* potrà essere assegnato ad un intero passando per la classe *wrapper Integer.*

UNBOXING E COLLECTION

Per estrarre un valore da un elemento di una collection, la sola tecnica di *unboxing* non presenta particolari vantaggi rispetto all'uso di una semplice classe *wrapper.* In pratica è comunque necessario effettuare il cast dell'elemento generico della collection nella classe *wrapper* opportuna; cosa che già facevamo con Java 1.4. Di seguito è mostrato un esempio:

```
ArrayList list = new ArrayList();
list.add(10);
list.add(5);
```

NOTA

AUTOBOXING E PASSAGGIO DEI PARAMETRI

Mediante la tecnica di autoboxing è possibile definire dei metodi che accettano parametri generici di tipo *Object* e successivamente invocarli passando dei valori appartenenti a tipi base. Consideriamo il seguente metodo:

```
public void foo(Object obj) { ...}
```

Con Java 1.5 sarà possibile invocare questo metodo passando un tipo base, per esempio il valore intero 5 oppure il valore decimale 5.4:

```
// La variabile a è un oggetto della
// classe che contiene foo
a.foo(5);
a.foo(5.4);
```

Ovviamente dovrà essere il metodo *foo* a far buon uso del valore passato in base al tipo di dato. Quello che avviene in questi casi è un autoboxing del parametro in un oggetto appartenente all'opportuna classe wrapper, ed il passaggio di un riferimento ad esso. Ciò spiega come sia stato possibile passare un intero al metodo add nella classe *ArrayList* in uno degli esempi precedenti.

```
list.add(3);
for (int i=0; i<list.size(); i++) {
  int j = (Integer) list.get(i); //Unboxing
  System.out.println("List("+i+") = " + j); }
```

Nel ciclo *for*, come si può notare, si estrae l'elemento intero dall'*ArrayList* utilizzando il metodo *get* e il *cast*. L'unica novità è rappresentata dal fatto che non è necessario invocare esplicitamente il metodo *intValue;* l'operatore = è sufficiente.

CLASSI WRAPPER ED OPERATORI

Quando si usa la tecnica di *autoboxing* per un dato tipo, l'oggetto appartenente alla classe *wrapper* supporta tutti gli operatori previsti per il tipo base.
Nell'esempio che segue, l'oggetto *k* verrà incrementato utilizzando l'operatore ++*:*

```
Integer k = 37;
k++;
System.out.println(k); // Visualizza 38
```

In questo esempio, a fronte dell'operazione ++, vengono eseguite le seguenti operazioni:

```
unboxing di k in un valore intero;
incremento del valore intero;
autoboxing del valore intero nell'oggetto k.
```

Di seguito, invece, utilizziamo l'operatore complemento su un oggetto di tipo *Boolean*:

```
Boolean bool = false;
System.out.println(!bool); // Visualizza true
```

Dagli esempi si evince che gli oggetti appartenenti alle classi *wrapper* possono essere coinvolti in espressioni al pari dei loro rispettivi tipi base.

CICLI

Al fine di scandire automaticamente gli elementi di un array o di una collection, l'istruzione *for*, oltre a quella tradizionale, presenta una nuova sintassi:

```
for (Tipo Identificatore : Espressione)
  Istruzioni
```

Espressione è un'istanza di un array oppure della nuova interfaccia *Iterable* (In Java 1.5 l'interfaccia *Collection* implementa *Iterable*); *Tipo* rappresenta il tipo di dato del singolo elemento all'interno dell'array o della collection; *Identificatore* è il nome assegnato a tale singolo elemento. Come tutti sappiamo, per scandire gli elementi di una array si procede in questo modo:

```
int v[]={4,6,87,9};
for (int i = 0; i < v.length; i++) {
  System.out.println(v[i]); }
```

Con Java 1.5 è ancora possibile utilizzare questa sintassi, anche se è consigliato usare la nuova versione del ciclo *for.* Applicando la nuova sintassi otterremo:

```
int v[]={4,6,87,9};
for (int item: v) {
  System.out.println(item);
}
```

Come si può notare, il codice risultante è più compatto. A prima vista può sembrare meno leggibile, tuttavia non è così. Il fatto è che siamo abituati al vecchio ciclo *for* e soprattutto all'indice. Come si può notare, usando il nuovo *for* non c'è traccia di indice. È la variabile *item*, definita all'interno delle parentesi, che assumerà ad ogni passaggio il valore corrispondente nell'array *v.* Quindi al primo ciclo *item* sarà 4, al secondo 6, poi 87 e infine 9. Se desideriamo effettuare la somma degli elementi presenti nell'array *v* possiamo procedere così:

```
// Somma
int v[]={4,6,87,9};
int somma = 0;
for (int item: v) { somma += item; }
System.out.println(somma);
```

Semplice, no? In altre tecnologie, come .NET oppure Visual Basic 6, un costrutto simile è rappresentato dall'istruzione *foreach* che funziona praticamente nello stesso modo del *for* "migliorato" di Java 1.5. Personalmente credo che la parola chiave *foreach* (per ogni) sia un tantino più chiara. Consideriamo il seguente codice C#:

```
// Codice C#
foreach (int item in v) {
   System.Console.WriteLine(item);
}
```

Leggendolo si ottiene qualcosa di molto simile al linguaggio naturale: *Per ogni (foreach) item intero nell'array v fai…* In Java, volutamente, non è stata introdotta una nuova parola chiave. In effetti, dopo nove anni, gli esperti della Sun hanno ritenuto non opportuno effettuare una modifica così importante, che avrebbe potuto mettere a rischio codice già esistente, si pensi al fatto che *foreach* potrebbe essere stato usato come identificatore. Quindi, come abbiamo visto, è stata affiancata all'istruzione *for* una nuova sintassi.

CICLO FOR E COLLECTION

Il nuovo ciclo *for* può funzionare anche con le collection ed in generale con tutte le classi che estendono o implementato la nuova interfaccia *java.lang.Iterable*. Nel seguente esempio è mostrato come scandire un *ArrayList* prima dell'avvento di Java 1.5:

```
ArrayList list = new ArrayList();
...
for (int i=0; i<list.size(); i++) {
  Integer k = (Integer) list.get(i);
  System.out.println(k);}
```

Utilizzando il nuovo *for* è possibile effettuare la medesima operazione nel seguente modo:

```
ArrayList list = new ArrayList();
...
for (Object obj: list) {
  Integer k = (Integer) obj;
  System.out.println(k); }
```

Come si può notare, il funzionamento per le collection è decisamente molto simile a quello visto per gli array. Come abbiamo detto, l'espressione all'interno del ciclo *for* deve essere un'istanza di una array oppure un'istanza della nuova interfaccia *Iterable*.
Questa nuova interfaccia è stata introdotta per evitare dipendenze del linguaggio con il package *java.util*. A rigor di logica, si sarebbe potuto progettare il nuovo *for* in modo che lavorasse direttamente con le istanze di tipo *Collection*. Questa però è un interfaccia presente in *java.util*. Procedendo in questo modo, una caratteristica propria del linguaggio, cioè l'istruzione *for*, sarebbe diventata dipendente da qualcosa di esterno al linguaggio (il package *java.util*).
Per evitare tale dipendenza è stata introdotta l'interfaccia *Iterable* in *java.lang* (quindi all'interno del linguaggio) in modo che il nuovo ciclo *for* lavori su istanze di essa. Inoltre, affinché *for* funzioni anche con le collection, l'interfaccia *Collection* in Java 1.5 estende l'interfaccia *Iterable*.

CICLI ANNIDATI

Vediamo adesso come, utilizzando il nuovo *for*, sarà possibile ottenere cicli annidati più leggibili ed eleganti. Supponiamo di avere l'ArrayList *giorni* in cui gli elementi sono a loro volta ArrayList contenti appuntamenti per tali giorni. Per visualizzare tutti gli appuntamenti per tutti i giorni è necessario annidare due cicli, come mostra il seguente frammento di codice:

```
for (int i=0; i<giorni.size(); i++) {
  ArrayList appuntamenti = (ArrayList) giorni.get(i);
  for (int j=0; j<appuntamenti.size(); j++) {
    System.out.println(appuntamenti.get(j));} }
```

Come si può notare, la leggibilità è molto scarsa e gli indici posso essere causa di confusione nonché di insidiosi bug. Riscrivendo lo stesso codice facendo uso di iteratori, la leggibilità non risulta migliorata:

```
for (Iterator it = giorni.iterator(); it.hasNext();) {
  ArrayList appuntamenti = (ArrayList) it.next();
  for (Iterator jt = appuntamenti.iterator();
    jt.hasNext();) {
    System.out.println(jt.next()); } }
```

Utilizzando la nuova sintassi per l'istruzione *for* di Java 1.5 e i *generics* (li incontreremo il prossimo mese) il risultato, in termini di leggibilità e compattezza, è veramente notevole:

```
for (ArrayList<String> appuntamenti: giorni)
    for (String nota: appuntamenti)
      System.out.println(nota);
```

All'interno del primo ciclo la variabile *giorni* sarà un ArrayList di stringhe (appuntamenti); all'interno del secondo *for* si scandisce questo ArrayList e per ogni elemento di esso, viene stampata la stringa nota.

TIPI GENERICI

L'obiettivo della programmazione generica è quello di costruire strutture dati, o più in generale classi, operanti su tipi di dato generici. Ciò significa che la struttura sarà implementata una sola volta, ma potrà essere utilizzata (quindi istanziata) più volte specificando ogni volta il tipo di dato coinvolto in essa. Per esempio, la struttura dati stack (pila), se scritta secondo il paradigma della programmazione generica, potrà operare su ogni tipo di dato consentito dal linguaggio di programmazione utilizzato.
Fino alla versione 1.4, la programmazione generica in Java poteva essere attuata, come vedremo nel prossimo paragrafo, esclusivamente attraverso la classe *Object* e le strutture dati basate su di essa. Con l'avvento di Java 1.5, è possibile invece perseguire una strada per certi aspetti simile ai template del C++. Stiamo parlando dei tipi generici, o *generics* come vengono denominati nel gergo Sun, che saranno l'argomento di questo articolo.

PROGRAMMAZIONE GENERICA IN JAVA 1.4

System.Object è la classe base di tutte le classi Java. Questo, in particolare, consente di assegnare ad

un'istanza di *Object* un riferimento a qualsiasi altra istanza di classe. Grazie ad essa è possibile attuare la programmazione generica. Vediamo come.
Supponiamo di voler implementare la struttura dati stack in modo da memorizzare qualsiasi tipo di elemento. Sebbene in *java.util* sia possibile trovare l'implementazione della classe *Stack*, per motivi didattici, ignoriamo la sua esistenza. Come saprete, questa struttura dati ammette solo due primitive: push e pop. La prima inserisce un elemento in cima dello *stack*, la seconda invece lo rimuove e lo restituisce. L'implementazione della classe *Stack* è la seguente:

```
class Stack {
  private ArrayList elements = new ArrayList();
  public void push(Object item) {
    elements.add(item);}
  public Object pop() {
    int top = elements.size()-1;
    Object item = elements.get(top);
    elements.remove(top);
    return item; }
}
```

Come si può notare, abbiamo utilizzato un *ArrayList* per memorizzare gli elementi ed implementato i metodi push e pop, che rappresentano le due primitive. Inoltre, Come si può notare dalle signature dei metodi, gli elementi passati e restituiti sono istanze della classe *Object*. Questo ci consente di utilizzare la classe *Stack* per memorizzare istanze di ogni tipo. La classe è quindi generica. Quello che segue è un esempio d'utilizzo di uno stack di stringhe.

```
Stack stack = new Stack();
stack.push("Valentina");
stack.push("Luisa");
String top = (String) stack.pop();
```

Sebbene attraverso questa strategia la programmazione generica è garantita, ci sono almeno tre svantaggi in questo modo d'operare:

1. Come sappiamo, un tipo base non è una classe e quindi non estende *System.Object*. Di conseguenza il nostro stack non funzionerà, almeno direttamente, con tipi base. Per ovviare a questo problema, è necessario utilizzare le classi wrapper. Nell'esempio che segue, utilizzeremo la classe *Stack* con valori interi:

   ```
   Stack stack = new Stack();
   stack.push(new Integer(8));
   stack.push(new Integer(32));
   int top = ((Integer)stack.pop()).intValue();
   ```

   Come si può notare, è necessario ricorrere alla classe wrapper *Integer*.
2. Ogni volta che un elemento viene prelevato è necessario un cast esplicito:

   ```
   String top = (String) stack.pop();
   ```

   Questo modo d'operare, non solo rallenta le prestazioni, ma rende l'applicazione particolarmente vulnerabile all'eccezione *ClassCastException*, che è uno delle principali fonti di bug per applicazioni Java.
3. Nessuno vieta di inserire in uno stesso stack oggetti di diverso tipo, come mostrato dal seguente esempio:

   ```
   stack.push(new Integer(8));
   stack.push("Ciao");
   ```

   Noterete che, nello stack sono stati inseriti una stringa ed un intero.
   In linea di principio, una cosa del genere non è coerente con la filosofia di base delle strutture dati classiche, secondo la quale, gli elementi dovrebbero essere omogenei. Problemi simili emergono ogni qualvolta si ricorre alla programmazione generica basata sulla classe *Object*. Per evitarli, in Java 1.5 sono stati introdotti i tipi generici. Vediamo di cosa si tratta.

ANTICIPAZIONE

Prima di introdurre la sintassi per definire ed utilizzare un tipo generico in Java 1.5, mostreremo qualche frammento di codice in modo da fissare le idee in termini di finalità e benefici.
La classe *java.util.Stack*, in Java 1.5, è una collection che utilizza tipi generici. Sarà quindi possibile utilizzarla nel seguente modo:

```
Stack<String> stack =
  new Stack<String>();
stack.push("Valentina");
stack.push("Luisa");
String top = stack.pop();
```

La parte più strana del codice appena visto è di sicuro l'istruzione:

```
Stack<String> stack =
  new Stack<String>();
```

Precedendo in questo modo, stiamo dichiarando che utilizzeremo l'istanza stack esclusivamente per ospitare stringhe. Ciò è possibile poiché, come detto in precedenza la classe *java.util.Stack*, utilizza un tipo generico, che viene successivamente specificato in fase di istanziazione. Nell'esempio, il tipo generi-

Sun Microsystems, JavaTM 2 SDK, Standard Edition, Version 1.5.0 - Summary of New Features and Enhancements.
http://java.sun.com/j2se/1.5.0/docs/relnotes/features.html

Calvin Austin, J2SE 1.5 in a Nutshell.
http://java.sun.com/developer/technicalArticles/releases/j2se15/

Sun Microsystems, JSR 201: Extending the JavaTM Programming Language with Enumerations, Autoboxing, Enhanced for loops and Static Import.
http://jcp.org/en/jsr/detail?id=201

co assumerà, per l'oggetto stack, il valore *String*.
È possibile istanziare la classe *Stack* in modo che funzioni anche con tipi base. Nel seguente esempio sarà utilizzato uno stack di interi.

```
Stack<Integer> stack =
  new Stack<Integer>();
stack.push(8);
stack.push(32);
int top = stack.pop();
```

Passare direttamente un *int* alla funzione *push* (e non un'istanza di *Integer*) è stato possibile grazie alla funzionalità di autoboxing. Utilizzando le classi generiche verranno risolti in uno solo colpo i tre problemi sollevati nella sezione precedente:

1. Si possono utilizzare tipi base (grazie all'autoboxing);
2. Non è necessario un cast per prelevare gli elementi;
3. È possibile specificare un singolo tipo di elemento in modo da non avere stack eterogenei.

Molte delle collection presenti in Java 1.5 lavorano su tipi generici. Nel seguente frammento di codice, per esempio, viene utilizzato un *ArrayList* di interi:

```
ArrayList<Integer> list
  = new ArrayList<Integer>();
list.add(26);
list.add(29);
list.add(18);
for (Integer item : list) {
  System.out.println(item);}
```

Ovviamente, il tipo generico potrà assumere come valore finale anche una classe definita dal programmatore. Se per esempio la classe *Persona* è la seguente:

```
class Persona {
  public String nome;
  public String cognome;
}
```

Allora sarà possibile istanziare un *ArrayList* contenente istanze di *Persona*, come mostrato di seguito:

```
ArrayList<Persona> list
  = new ArrayList<Persona>();
Persona p = new Persona();
p.nome = "Mario";
p.cognome = "Rossi";
list.add(p);
p = new Persona();
p.nome = "Antonio";
p.cognome = "Bianchi";
list.add(p);
for (Persona item : list) {
  System.out.println(item.nome +
    " " + item.cognome);
}
```

CLASSI CHE USANO TIPI GENERICI

Per implementare una classe che usa un tipo generico è necessario usare una nuova notazione:

```
class nome-classe<etichetta-tipo-generico>
```

dove etichetta-tipo-generico sarà usata all'interno della classe per operare sul tipo generico e sarà sostituita con uno tipo concreto in fase di istanziazione. Supponendo di voler implementare la classe Stack in modo generico, bisogna procedere nel seguente modo:

```
class Stack<E> {
  private ArrayList<E>
    elements = new ArrayList<E>();
  public void push(E item) {
    elements.add(item); }
  public E pop() {
    int top = elements.size()-1;
    E item = elements.get(top);
    elements.remove(top);
    return item; }
}
```

Come si può notare, l'etichetta usata per rappresentare il tipo generico è E. Ciò vuol dire che:

- la classe *Stack* utilizza un *ArrayList* di *E*;
- il metodo push prende in input un elemento di tipo *E*;
- il metodo pop restituisce un elemento di tipo *E*.

Nel momento in cui desideriamo istanziare la classe *Stack*, il tipo *E* potrà essere specificato con un tipo reale. Nell'esempio che segue utilizzeremo la classe *String:*

```
Stack<String> stack =
  new Stack<String>();
stack.push("Valentina");
stack.push("Luisa");
String top = stack.pop();
```

In questo secondo esempio utilizzeremo la classe *Stack* in modo che operi su oggetti di tipo *Persona:*

```
Stack<Persona> stack =
  new Stack<Persona>();
```

K. Kreft, A. Langer, Language Features of Java Generics, JavaPro Online, Marzo 2004.
www.ftponline.com/javapro/2004_03/online/jgen_kkreft_03_03_04/

E. E. Allen, Diagnosing Java code: Java generics without the pain (Parts I-IV)
www-106.ibm.com/developerworks/java/library/j-djc02113.html

```
Persona p = new Persona();
p.nome = "Mario";
p.cognome = "Rossi";
stack.push(p);
p = new Persona();
p.nome = "Antonio";
p.cognome = "Bianchi";
stack.push(p);
Persona top = stack.pop();
```

Nel caso in cui il tipo *E* non venga specificato, il compilatore assumerà *System.Object:*

```
Stack stack = new Stack();
stack.push("Valentina");
stack.push("Luisa");
String top = (String) stack.pop();
```

Nome	Tipologia
ArrayList<E>	classe
Collection<E>	interfaccia
Comparable<T>	interfaccia
Comparator<A>	interfaccia
Dictionary<K,V>	classe
Enumeration<E>	interfaccia
HashMap<K,V>	classe
HashSet<E>	classe
Hashtable<K,V>	classe
Iterator<E>	interfaccia
LinkedList<E>	classe
List<E>	interfaccia
ListIterator<E>	interfaccia
Map<K,V>	interfaccia
Set<E>	interfaccia
Stack<E>	classe
TreeMap<K,V>	classe
TreeSet<E>	classe
Vector<E>	classe

Tabella 1: Classi e interfacce

Nell'esempio precedente lo stack conterrà oggetti generici, questo spiega la necessità del cast nell'effettuare il *pop*.
È possibile anche parametrizzare una classe utilizzando due o più tipi generici. Di seguito riportiamo un'implementazione parziale di un hash table:

```
class Hashtable<K,V> {
  public void put(K key, V value) {
    ...
    ... }
  public V get(K key) {
    ...
    ... }
}
```

Come si può notare, il primo tipo *(K)* è riferito alla chiave, il secondo *(V)* al valore. Se si desidera istanziare una hash table in cui le chiavi siano interi e i valori stringhe, si dovrà procedere nel seguente modo:

```
Hashtable<Integer, String> h =
  new Hashtable<Integer, String>();
h.put(1,"Valentina");
h.put(2,"Luisa");
String item = h.get(1);
```

Anche le interfacce possono usare tipi generici; di seguito ne riportiamo un esempio:

```
interface Queue<E> {
  public void add(E item);
  public E rear(); }
```

L'interfaccia definisce le primitive *add* e *rear* della struttura dati *queue* (coda) che operano su un tipo generico *E*. L'interfaccia può essere implementata nel seguente modo:

```
class RealQueue<E> implements Queue<E> {
  public void add(E item) {
    ...
    ... }
  public E rear() {
    ...
    ... }}
```

Come si può notare, l'implementazione *RealQueue* utilizzerà anch'essa lo stesso tipo generico *E*. I tipi generici sono estremamente utili quando utilizzati con le collection. Per questo motivo, in Java 1.5 le varie implementazioni delle collection sono state riscritte utilizzando tipi generici. Nella seguente tabella sono mostrate le classi e le interfacce più comuni di Java 1.5, che utilizzano tipi generici (Tab. 1).

COSE DA EVITARE

Ci sono delle limitazioni nell'utilizzo dei tipi generici. La maggior parte di queste esistono poiché il comitato di definizione di Java 1.5 ha stabilito di modificare il compilatore per supportare i tipi generici, ma non la Virtual Machine. Se da una parte questo rende possibile eseguire applicazioni Java scritte con la versione 1.5 anche su Virtual Machine precedenti, dall'altra tale decisione porta inevitabili limitazioni sia dal punto di vista dello sviluppo sia da quello relativo alle performance. In particolare non è consentito (oppure se lo è non è consigliato) effettuare le seguenti operazioni:

- usare il tipo generico per dati membro statici;
- usare tipi base come tipi generici;
- un tipo generico secco non dovrebbe essere usato in operazioni instanceof;
- un tipo generico secco non dovrebbe essere usato in operazioni new;
- un tipo generico secco non dovrebbe essere usato in operazioni implements o extends.

Per tipo generico secco si intende l'etichetta del parametro. Per intenderci: *E* è un tipo generico secco, *ArrayList<E>* no. Inoltre, usando i tipo generici, non si ha nessun beneficio dal punto di vista dell'esecuzione. Ciò accade perché nel momento in cui si compila una classe che usa tipi generici, il compilatore rimuove tutte le etichette assegnate ai parametri e le sostituisce con la classe *Object* aggiungendo tutti i cast necessari che il programmatore non ha scritto. In pratica ciò che i tipi generici ci danno è una sintassi migliore, che probabilmente farà risparmiare del tempo in fase di sviluppo, renderà il codice più leggibile, ma non porterà nessun beneficio

NOTA

Altre novità molto importanti di Java 1.5 sono le enumerazioni e le liste variabili di argomenti. In quest'articolo esamineremo entrambi gli aspetti mostrandone esempi concreti d'utilizzo.

dal punto di vista delle prestazioni. Un altro problema, ancora più grave forse, è rappresentato dal fatto che fondamentalmente è stata introdotta una discrepanza tra ciò che si ha in fase di compilazione e quello che in effetti va in esecuzione. Ciò accade perché il tipo generico viene sostituito con la classe *Object*. Quindi, se definiamo un ArrayList di Integer, in fase di run-time ci ritroveremo un ArrayList di *Object*. Ci si accorge di tale discrepanza nel momento in cui si utilizza la Reflection per ispezionare classi che usano istanze di classi generiche. Per esempio una List di List a run-time è semplicemente una lista di oggetti; neanche ispezionandola attraverso la Reflection sarà possibile scoprire che in realtà ogni elemento della *List* è a sua volta un'altra istanza di *List*.

COSTANTI STATICHE

Le enumerazioni (enum), in determinati casi, sono un'alternativa elegante alle costanti statiche. Supponiamo di avere la classe *TShirt* e di voler definire il campo taglia in modo che possa assumere i seguenti valori: *XS, S, M, L, XL, XXL*. La prima soluzione che ci viene in mente è quella di inserire il campo numerico taglia e definire delle costanti mnemoniche per facilitarne l'utilizzo. Ecco come possiamo definire la nostra classe *TShirt:*

```
class TShirt {
  public final static int XS = 0;
  public final static int S = 1;
  public final static int M = 2;
  public final static int L = 3;
  public final static int XL = 4;
  public final static int XXL = 5;
  private int taglia;
  public int getTaglia() {
      return taglia; }
  public void setTaglia(int taglia) {
      this.taglia = taglia; }
}
```

Supponendo di voler assegnare ad un'istanza di *TShirt* la taglia *XL*, possiamo procedere nel modo seguente:

```
// tshirt è un oggetto di tipo TShirt
tshirt.setTaglia(TShirt.XL);
```

Come si può vedere, è stata utilizzata la costante *TShirt.XL* piuttosto che il relativo valore intero. Ciò rende il codice più leggibile e manutenibile. Ci sono però almeno due problemi con questo modo di operare:

1. Non esiste alcun legame effettivo tra il campo *taglia* e le costanti; queste, essendo dei semplici valori interi, potrebbero essere utilizzate impropriamente in altri contesti; di seguito, ne riportiamo un esempio di errato utilizzo:

```
int età = TShirt.XL;
```

2. Al campo *taglia* potrebbe essere assegnato un valore fuori range, per esempio 150:

```
tshirt.setTaglia(150);
```

Per questo, ed altri motivi, in Java 1.5 sono state introdotte le enumerazioni.

ENUM

Un'enumerazione è un tipo di dato definito dall'utente che presenta la seguente sintassi:

```
[modificatore-d'accesso] enum {lista-valori}
```

Applichiamo questo nuovo concetto all'esempio precedente. L'enumerazione relativa al campo taglia può essere definita come segue:

```
enum Taglia {XS, S, M, L, XL, XXL }
```

Quindi, la classe *TShirt* utilizzerà l'enum *Taglia* come tipo per il campo taglia:

```
class TShirt {
  private Taglia taglia;
  public Taglia getTaglia() {
      return taglia; }
  public void setTaglia(Taglia taglia) {
      this.taglia = taglia; } }
```

Per assegnare un valore al campo taglia, si procede in questo modo:

```
tshirt.setTaglia(Taglia.XL);
```

È importante notare che, utilizzando le enumerazioni, non è più possibile eseguire un'istruzione del tipo:

```
// Istruzione errata
tshirt.setTaglia(150);
```

Inoltre, non sarà consentita neanche l'operazione inversa:

```
// Istruzione errata
int a = tshirt.getTaglia();
```

I problemi sollevati nel paragrafo precedente, sono stati quindi risolti. Le enumerazioni vengono visua-

ENUMERAZIONI E ARGOMENTI

Variabili
Jesse Liberty, autore di Programming C#, accenna al fatto che è quasi inspiegabile che un aspetto così importante come le enumerazioni sia stato lasciato di proposito fuori dal linguaggio Java. Molti, io compreso, concordano con Jesse. Anche qualcuno in Sun deve essere stato d'accordo con lui, visto che, come esamineremo in questo articolo, Java 1.5 finalmente le supporterà.

lizzate sulla console con il nome simbolico. Per esempio l'istruzione:

```
System.out.println("Taglia = " + Taglia.XL);
```

visualizzerà *XL*. Per ottenere l'elenco di tutti i valori presenti in un enum, si deve utilizzare il metodo statico values. Nell'esempio che segue saranno visualizzate tutte le costanti dell'enum Taglia:

```
for (Taglia t : Taglia.values())
  System.out.println(t);
```

Il metodo statico *values*, generato automaticamente dal compilatore, restituisce un oggetto di tipo List contenente tutte le costanti previste per l'enum in questione.

VALORI PER LE COSTANTI

Molto spesso è necessario associare ad un'etichetta di un enumerazione un valore ben definito. Cerchiamo di spigare il concetto con un esempio. Prima della fine del XX secolo (nonché del II millennio) a Roma esisteva un convenzione molto diffusa per riferirsi alla moneta dell'epoca: la lira. Nella tabella seguente mostriamo alcune delle locuzioni usate per indicare una certa quantità di denaro (Tab 2). Supponiamo di voler definire un'enumerazione che tenga conto della denominazione romana del XX secolo e del rispettivo valore. Con Java 1.5 possiamo definirla nel seguente modo:

```
public enum ValutaRomana {
  SACCO(1000),
  SCUDO(5000),
  PIOTTA(100000),
  TESTONE(1000000);
  private final int value;
  private ValutaRomana(int value) {
    this.value = value; }
  public int value() {
    return value;
  }
}
```

Come si può notare, ad ogni costante è associato il rispettivo valore mediante la notazione:

```
COSTANTE(valore)
```

Ma c'è di più. L'enum presenta anche un costruttore privato che accetta il valore intero e lo memorizza nella variabile privata *value*. Esso verrà richiamato nel momento in cui si effettuerà un'assegnazione, del tipo:

```
ValutaRomana v = ValutaRomana.SACCO;
```

Ciò che avviene dietro le quinte, è la creazione dell'enum *v* mediante il costruttore privato che avrà in input 1000, ovvero il valore associato all'etichetta SACCO. Il metodo value restituisce il valore associato all'etichetta. Nell'esempio che segue saranno visualizzate tutte le denominazioni romane ed i rispettivi valori:

```
for (ValutaRomana v : ValutaRomana.values() )
  System.out.println(v + ": " + v.value());
```

L'output sarà quello di Fig. 1.

Fig. 1: ***Un esempio di utilzzo delle costanti***

Con l'avvento dell'euro, questa pittoresca convenzione è stata modificata, tuttavia siamo ancora ad una versione beta dell'euro convertitore. Secondo alcuni un sacco adesso vale 1 Euro, secondo altri no. Qualcuno ha perfino proposto una nuova nomenclatura per cui un eurosacco equivale ad 1 Euro, un euroscudo a 5 Euro e così via...

TIPO DELLE COSTANTI

È possibile utilizzare un tipo diverso da *int* per le costanti di un'enumerazione; basta specificare valori appropriati nella definizione della costante ed implementare il costruttore privato in modo opportuno. Nel seguente esempio viene definita l'enum Switch, che contiene due costanti booleane *ON* e *OFF*:

```
public enum Switch {
  ON(true),
  OFF(false);
  private final boolean value;
  Switch(boolean value) {
    this.value = value;  }
  public boolean value() {
    return value;  }}
```

Come si può notare, alla costante *ON* è associato il valore booleano true mentre alla costante *OFF* il valore *false*. In questo caso il costruttore avrà un valore

booleano come parametro. Il seguente frammento di codice visualizzerà l'output di Fig.2

```
Switch s = Switch.OFF;
System.out.println(s + "(" + s.value() + ")");
```

Fig. 2: ***Lo switch in azione***

COSTANTI ASSOCIATE A PIÙ TIPI

Una costante in un enum può anche essere anche associata a più tipi. Ritorniamo all'esempio dell'enumerazione *ValutaRomana* e supponiamo di voler associare, per ogni denominazione, anche una descrizione. Si dovrà procedere nel seguente modo:

```
public enum ValutaRomana {
  SACCO(1000,"'n sacco"),
  SCUDO(5000,"'no scudo"),
  PIOTTA(100000,"'na piotta"),
  TESTONE(1000000,"'n testone");
  private final int value;
  private final String description;
  private ValutaRomana(int value,
    String description) {
    this.value = value;
    this.description = description; }
  public int value() {
    return value;
  }
  public String description() {
    return description; }
}
```

Le modifiche, rispetto alla versione precedente, sono sostanzialmente tre:

1. La costante oltre ad avere un intero (il valore) presenta anche una stringa (la descrizione), che descrive il valore.

   ```
   SACCO(1000,"'n sacco")
   ```

   Nell'esempio, alla costante *SACCO* è associato il valore *1000* e la descrizione *"'n sacco"*, ovvero un sacco.
2. Il costruttore privato adesso accetterà due parametri: *value* (valore) e *description* (descrizione).
3. Il nuovo metodo description restituirà la descrizione.

Nel seguente esempio saranno visualizzate costanti, valori e descrizione per l'enum *ValutaRomana:*

```
for (ValutaRomana v : ValutaRomana.values() )
  System.out.println( v + " " + v.value + " " +
                                          v.description());
```

L'output è quello di Fig.3.

ENUM E SWITCH

Le enumerazioni possono essere utilizzate anche all'interno di istruzioni *switch*, come mostrato dal seguente esempio:

```
for (Taglia t : Taglia.values() ) {
  switch (t) {
    case XS:
      System.out.println("La taglia è XS");
      break;
    case S:
      System.out.println("La taglia è S");
      break;
    case M:
      System.out.println("La taglia è M");
      break;
    case L:
      System.out.println("La taglia è L");
      break;
    case XL:
      System.out.println("La taglia è XL");
      break;
    case XXL:
      System.out.println("La taglia è XXL");
      break;
  }
}
```

Fig. 3: ***Diversi tipi associati alle costanti***

Questa è una caratteristica che risulterà particolarmente utile in numerosi contesti.

CONCLUSIONI

Come si è visto, le enumerazioni sono un aspetto molto importante della programmazione orientata agli oggetti; è un grosso vantaggio che anche Java le supporti. Inoltre, la possibilità di definire metodi che presentino come parametro una lista variabile d'argomenti è un aspetto che risulterà sicuramente molto utile.

Giuseppe Naccarato

Progettiamo un sistema che "alimenta" un feed RSS

La pubblicazione di articoli con RSS

Sempre più siti pubblicano online dei file XML chiamati feed RSS, che permettono agli utenti di essere automaticamente aggiornati sulle novità. Creiamo con .NET un sistema che pubblica un feed RSS

I siti di notizie che ci interessano, i blog che leggiamo, i forum che seguiamo sono sempre più numerosi. Spesso per vedere se c'è qualcosa di nuovo dobbiamo fare il giro di tutti, anche se spesso non troviamo niente di nuovo o di interessante. Oppure vogliamo fare una sorta di rassegna stampa degli ultimi articoli pescati dalla rete: dobbiamo andare a cercarli, copiare, incollare… Per fortuna non è il solo metodo. Sempre più siti pubblicano online dei file XML chiamati feed RSS che permettono a chi segue il sito di essere automaticamente aggiornato sulle novità. Li riconosciamo spesso per un simbolino arancione o nero, ormai standard (Fig. 1). Questi feed RSS altro non sono che file XML prodotti secondo una metodologia standard che è possibile leggere ma soprattutto dare in pasto a dei programmi che li possano facilmente decodificare per noi (ah! L'XML, come faremmo senza?). Questi programmi si chiamano aggregatori, e permettono di tenersi sempre automaticamente aggiornati sulle notizie pubblicate dai nostri siti preferiti. Non dobbiamo pensare però solo da fruitori dei feed RSS. Da bravi sviluppatori potremmo interessarci anche alla produzione di feed, per esempio per notificare ai visitatori del nostro sito la presenza di nuovi articoli. Con Microsoft Word 2003 è stato introdotto un nuovo standard XML di definizione del documento chiamato WordML. Questo standard ci permette di effettuare un parsing del documento in modo piuttosto agevole, cosa impossibile o quasi con formato Word binario. Uniamo queste due tecnologie con la potenza di .NET per creare un sistema che pubblichi sul web un feed RSS per la notifica della presenza di articoli tecnici sul nostro sito.

Fig. 1: *Vari simboli dei feed RSS*

COSA IMPAREREMO? A COSA CI SERVIRÀ REALIZZARE QUESTO PROGETTO?

Impareremo cosa sono i feed RSS e come sono fatti. Impareremo a produrre una classe a partire da uno schema XSD. Impareremo qualcosa sullo sterminato mondo della Serializzazione in .NET, vedremo come costruire una classe che faccia il parsing di un documento WordML e la realizzeremo con un approccio modulare. Poi scriveremo il codice che verificherà la pubblicazione di nuovi articoli Word, li esaminerà e pubblicherà sul Web un feed RSS aggiornato. Per vedere tutte queste cose avremo bisogno di due puntate. In questa stessa ci occuperemo della produzione del feed RSS, nella prossima completeremo il progetto.

COSA È UN FEED RSS

Vediamo più da vicino la natura del file che pubblicheremo. RSS è una sigla che sta per molte cose: *RDF Site Summary, Rich Site Summary, Really Simple Syndication.* A cosa serve, lo abbiamo capito: è un formato XML che permette la pubblicazione di un elenco di notizie, o canali, in modo che sia facilmente interpretabile dalle macchine. Vediamo più da vicino come è fatto un feed. Semplicemente sarà un file XML, con dentro delle informazioni codificate in maniera standard. Tanto standard, che ne esistono (ad oggi) quattro versioni: 0.91, 0.92, 1.0 e 2.0.

Nell'esempio ci occuperemo della 2.0, la più nuova, retrocompatibile con 0.91 e 0.92. Va notato comunque che utilizzeremo soltanto un sottoinsieme degli elementi disponibili, quindi non sarà difficile convertire il nostro feed alla versione che preferiamo. Il lavoro che produrremo sarà infatti una base, sulla quale potrete lavorare per aggiungere le caratteristiche che più vi sembreranno utili, come i moduli RSS che ne estendono il modello. I quattro principali elementi del file RSS sono quelli riportati in *Tabella 1* e in forma d'albero, in Fig.2.

UNA GERARCHIA DI CLASSI CHE REALIZZA L'RSS

Vediamo come scrivere una gerarchia di classi C# che ci aiuti a produrre il feed. Perché costruirci delle classi? Chiaramente potremmo utilizzare direttamente i servizi del namespace *System.XML* che ci fornisce tutti i mezzi per produrre XML, non sarebbe una soluzione da disprezzare. Prenderemo però un'altra strada, per due motivi:

1) potremo operare in maniera Object Oriented direttamente sulla struttura logica del feed
2) quello che vedremo ci permetterà di imparare qualcosa di nuovo: la produzione di classi a partire dallo schema XSD e l'uso della serializzazione.

La tecnica che utilizzeremo si basa sull'uso del programma *xsd.exe*, che potete trovare nella cartella *C:\Programmi\Microsoft Visual Studio .NET 2003\SDK\v1.1\bin* o equivalente, nel caso non abbiate Visual Studio ma solo l'SDK. *Xsd.exe* è una utilità che permette di generare una classe dato uno schema, appunto, *xsd*. Sicuramente avete già sfruttato i suoi servigi, infatti viene chiamato dietro le quinte da Visual Studio ogni qual volta viene generato un Dataset tipizzato. Tra le opzioni del programma, infatti, è possibile specificare se in uscita vogliamo una classe che realizzi un Dataset o al contrario una classe semplicemente serializzabile in XML. Ricordo che una classe è serializzabile quando è possibile salvare e ripristinare lo stato di un oggetto istanza della classe stessa. Più semplicemente quando possiamo "salvare" un oggetto, distruggerlo, crearne un altro e "caricare" l'oggetto precedente (ovvero il suo stato) in quest'ultimo. Infatti la cosa più interessante di *xsd.exe* è proprio la capacità di produrre classi che realizzano, se serializzate, un file XML compatibile con l'*xsd* di partenza. È opportuno notare che *xsd .exe* può essere utilizzato anche per fare l'inverso, ovvero geneare schemi a partire da tipi contenuti in un assembly.

FACCIAMO UN ESEMPIO

Questa è una tecnica che potremo utlizzare in tutti i nostri progetti. Facciamo quindi un semplice esempio, senza per ora complicarci la vita con l'RSS, per capirla meglio. Supponiamo di voler creare una classe che realizzi per noi un file xml del tipo

```
<?xml version="1.0"?>
<ordini>
<ordine id="ordine1" cliente="superpippo" >
  <dettaglio id="76-as" quantita="387" costo="34"/>
  <dettaglio id="23-bb" quantita="3" costo="123"/>
  <dettaglio id="mpmv" quantita="89"  costo="52"/>
  </ordine>
</ordini>
```

Elemento	Descrizione	
rss	Il nodo di root del documento	
channel	Contiene l'intero feed, I suoi contenuti e I suoi metadata	
image	Sottoelemento di channel che contiene dati sull'icona associata al feed	
item	Ogni canale può contenere più item. Ogni item definisce un'unità di contenuto, ed è l'elemento più importante. Contiene, tra gli altri	
	title	il titolo dell'articolo
	link	la url dell'articolo
	description	una descrizione testuale (no HTML) dell'articolo

TABELLA 1: Elementi base di RSS.

Uno schema possibile potrebbe essere questo generato da *xsd.exe* a partire dal documento xml visto prima, per cui sono presenti attributi e namespace Microsoft. Chiaramente qualunque *xsd* valido può essere utilizzato per i nostri scopi.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="ordini" xmlns=""xmlns:xs=
 "http://www.w3.org/2001/XMLSchema" xmlns:msdata
        ="urn:schemas-microsoft-com:xml-msdata">
 <xs:element name="ordini" msdata:IsDataSet=
                    "true" msdata:Locale="it-IT">
  <xs:complexType>
   <xs:choice maxOccurs="unbounded">
    <xs:element name="ordine">
     <xs:complexType>
      <xs:sequence>
...
```

Una volta ottenuto lo schema, sarà banale ottenere una classe C# che ne deriva: basterà accedere al prompt dei comandi, accertarsi che *xsd.exe* sia nel path, e digitare il comando (il significato delle opzioni è riportato nel box a fianco):

```
xsd.exe /c /n:TestXSD  /o: <directory di uscita>
                          nome_del_file_schema.xsd
```

Otterremo un file *nome_del_file_schema.cs* che conterrà la definizione di tre classi: *ordini, ordiniOrdine*

I FILE NEL CD

**• rss20.xsd: lo schema utilizzato per questo articolo.
È uno schema semplificato, non adatto per la verifica formale dei feed, ma sufficiente per generare le classi volute**

• rss20.cs: un esempio di classi generate.

BIBLIOGRAFIA

- CONTENT SYNDICATION WITH RSS *Ben Hammersley* (O'Reilly) 2003

e *ordiniOrdineDettaglio:*

```
namespace TestXSD {
  using System.Xml.Serialization;
  /// <remarks/>
  [System.Xml.Serialization.XmlRootAttribute(
                              Namespace="", IsNullable=false)]
  public class ordini {
    /// <remarks/>
    [System.Xml.Serialization.XmlElementAttribute(
     "ordine", Form=System.Xml.Schema.XmlSchemaForm
                                          .Unqualified)]
...
```

È evidente che è stata generata una gerarchia di classi modellata sugli elementi e gli attributi specificati nello schema. A completare classi e proprietà troviamo degli attributi facenti riferimento al namespace *System.Xml.Serialization*. Questi attributi aiuteranno l'XmlSerializer che utilizzeremo più avanti nel serializzare in maniera corretta il documento, specificando varie opzioni come il datatype XML da utilizzare o il nome dell'elemento serializzato, qualora ve ne fosse il bisogno. Vediamo come utilizzare tutto questo nel nostro codice. Per prima cosa inizializziamo l'oggetto principale, e poniamo per semplicità di volere un solo ordine:

```
using System.Xml.Serialization
TestXSD.ordini Doc= new TestXSD.ordini();
Doc.Items= new TestXSD.ordiniOrdine[1];
```

Poi andiamo a creare un ordine, che avrà due dettagli

```
TestXSD.ordiniOrdine Ordine= new TestXSD.ordiniOrdine();
Ordine.dettaglio=new TestXSD.ordiniOrdineDettaglio[2];
Ordine.cliente="superpippo";
Ordine.id="23";
```

A questo punto possiamo creare e aggiungere i dettagli all'ordine…

```
TestXSD.ordiniOrdineDettaglio Dettaglio= new
                          TestXSD.ordiniOrdineDettaglio();
Dettaglio.costo="988";
Dettaglio.id="tuta";
Dettaglio.quantita ="1";
Ordine.dettaglio[0]=Dettaglio;
...
```

… e l'ordine al documento

```
Doc.Items[0]=Ordine;
```

A questo punto costruiamo il serializzatore, che si preoccuperà di creare il nostro file XML tramite uno *StringWriter:*

```
XmlSerializer Ser= new
                  XmlSerializer(typeof(TestXSD.ordini));
System.IO.StringWriter SW= new
                              System.IO.StringWriter();
Ser.Serialize(SW,Doc);
```

Per vedere il nostro file XML sarà sufficiente utilizzare

```
SW.ToString();
```

che produce automaticamente quello che volevamo:

```
<?xml version="1.0" encoding="utf-16"?>
<ordini xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ordine id="23" cliente="superpippo">
    <dettaglio id="tuta" quantita="1" costo="988" />
    <dettaglio id="noccioline" quantita="300" costo="2" />
  </ordine>
</ordini>
```

È interessante notare tre cose:

1) Abbiamo utilizzato un approccio totalmente ad

NOTA

COMANDI DI XSD.EXE

```
xsd.exe <schema>.xsd  /classes|
    dataset  [/e:] [/l:] [/n:] [/o:] [/uri:]
xsd.exe <assembly>.dll|.exe
                [/outputdir:] [/type: [...]]
xsd.exe <instance>.xml [/outputdir:]
xsd.exe <schema>.xdr [/outputdir:]
```

In cui:

<schema>.xsd
nome dello schema che contiene gli elementi da importare

<assembly>.dll|exe
nome di un assembly i cui tipi devono essere convertiti in schemi

<instance>.xml
nome di un file xml da cui derivare un xsd

<schema>.xdr
nome di uno schema xdr da convertire in xsd

Opzioni:

/classes
/c
genera classi per lo schema fornito

/dataset
/d
genera dataset per lo schema fornito

/element:<element>
/e:
elemento da processare per lo schema

/language:<language>
/l:
il linguaggio con cui verrà generato il file di uscita. Può essere *"CS"* per C# (default), *"VB"* per Visual Basic.NET, *"JS"*, *"VJS"* o qualunque classe che implementa *System.CodeDom.Compiler.CodeDomProvider*

/namespace:<namespace>
/n:
il *namespace* della classe di uscita

/out:<directoryName>
/o:
la directrory dove creare il file. Se non viene specificata p la directory corrente.

/uri:<uri>
/u:
uri degli elementi dello schema da processare

oggetti per la produzione dell'XML. Immaginate come questo renderebbe semplice e pulito il nostro codice nel caso precedente se dovessimo creare un gran numero di dettagli e di ordini.

2) Se avessimo dovuto produrre un file invece che una stringa, sarebbe bastato scrivere

```
System.IO.FileStream FS=System.IO.File.Open(
            FileName,System.IO.FileMode.Create  );
Ser.Serialize (FS,RSS);
FS.Close();
```

3) Se avessimo avuto bisogno di specificare un altro tipo per gli attributi, per esempio un intero per la quantità, lo avremmo specificato sull'XSD:

```
<xs:attribute name="quantita" type="xs:int" />
```

che avrebbe fatto generare, nel file *.cs:*

```
[System.Xml.Serialization.XmlAttributeAttribute()]
public int quantita;
```

OK, COMINCIAMO!

Tornando al progetto originario, il nostro compito sarà:

1) ricavare un XSD che definisca la porzione delle specifiche RSS che ci interessano
2) ricavare con *xsd.exe* una classe C# che sia l'implementazione del file RSS e che sia facilmente serializzabile in un documento XML
3) valorizzare le proprietà della classe secondo i nostri contenuti
4) serializzare la classe nel nostro feed RSS

Lo standard RSS non è definito tramite un XSD, ma attraverso la descrizione degli elementi che lo compongono. Non è difficile però ricavare uno schema semplificato, che serva per i nostri scopi. Dovremo realizzare un file che abbia almeno gli elementi minimi di un feed, come nello lo stralcio seguente:

```
<?xml version="1.0"?>
<rss xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                                instance" version="2.0">
 <channel>
   <title>dotnet.innovactive.it categ. [Tutto]</title>
   <link>http://dotnet.innovactive.it/</link>
   <description>Il nuovo sito tecnico di
                               InnovActive</description>
...
```

Avremo quindi bisogno di un elemento *<rss>* che sia la root del nostro documento, poi una sequenza di *<channel>* e di *<item>* che contenga gli elementi che abbiamo visto prima e così via. Uno stralcio del file XSD che potremmo considerare è il seguente:

```
<xs:complexType name="channelClass">
  <xs:sequence>
    <xs:element name="title" type="xs:string" />
    <xs:element name="link" type="xs:anyURI" />
    <xs:element name="description" type="xs:string" />
    <xs:element name="language"
             type="xs:language" minOccurs="0" />
    <xs:element name="copyright" type=
                "xs:string" minOccurs="0" />
    ...
```

dove vediamo definiti gli elementi *channelClass* e *itemClass* che andranno a definire le classi che rappresenteranno channel ed item, così come tutti gli altri elementi *xClass*. Sul cd/sito troverete il file *.xsd* definitivo che verrà utilizzato nel progetto. Cercando su Internet comunque si trovano ottimi schemi, realizzati da sviluppatori, per qualunque versione dello standard. Come abbiamo visto è possibile utilizzare *xsd.exe* anche per derivare uno schema da un file XML di istanza, ma questa è un'opzione da prendere con cautela, infatti non è sempre possibile operare questa operazione di ingegneria inversa con successo. Il file *.cs* che otterremo sarà molto più complesso di quello del semplice esempio degli ordini visto prima, ma anche molto potente. Esaminandolo si vede come gli attributi di aiuto all'*XmlSerializer* specifichino il tipo degli elementi con maggiore dettaglio, ad esempio:

```
[System.Xml.Serialization.XmlArrayItemAttribute("hour
  ", DataType="nonNegativeInteger", IsNullable=false)]
public string[] skipHours;
```

PUBBLICHIAMO IL FEED

Utilizziamo le classi ora generate per pubblicare il nostro primo feed! Per far questo dovremo creare una semplice applicazione Web C# (o aggiungere una webform ad una applicazione già esistente) e far sì che l'XML creato venga servito all'utente. Abbiamo due possibilità: la prima è quella di creare al volo il feed e inviarlo direttamente all'utente ogni volta che questi fa una chiamata, l'altra è quello di generarlo soltanto quando ci sono aggiornamenti, salvarlo in un file, e mettere sul nostro sito un link al file. Nel primo caso avremmo un maggior carico sul server ma non ci dovremmo preoccupare di aggiornare alcunché; nel secondo caso il carico sarebbe molto ridotto, visto che verrebbe richiesto un file statico, ma dovremmo ricordarci di aggiornare il file ad ogni aggiunta di contenuto. Vediamo più in det-

www.syndic8.com

specifiche RSS
http://web.resource.org/rss/1.0/
http://backend.userland.com/rss
http://backend.userland.com/rss09
http://backend.userland.com/rss091
http://purl.org/rss/1.0/modules

un validatore di feed:
http://feedvalidator.org

aggregtaori gratuiti scritti con C#:
www.rssbandit.org
www.sharpreader.com

taglio il codice che potremo utilizzare per il primo caso. Per prima cosa creeremo un oggetto di classe *rssClass* che andrà a costituire l'elemento *RSS*, e uno di classe *channelClass* che andrà a costituire l'elemento *channel*. Poi andremo a valorizzare tutti gli attributi che intendiamo utilizzare.
Nell'esempio riportato si vede anche come venga inizializzata l'immagine rappresentativa del canale:

```
using RSS20;
using System.Xml.Serialization;
(...)
rssClass RSS= new rssClass();
RSS.channel= new channelClass();
...
```

e così via per tutti gli attributi che ci interessano. Una volta creato il channel passeremo all'aggiunta degli item.
In questo esempio inseriremo direttamente i valori, più realisticamente questi verranno presi da un repository esterno, come un database:

```
RSS.channel.item = new itemClass [2];
RSS.channel.item[0]= new itemClass();
RSS.channel.item[0].title="in bocca al lupo Elena";
RSS.channel.item[0].author="marco";
RSS.channel.item[0].description="torna presto!
                                  Diventa ricca!";
...
```

siamo pronti ad ottenere nel modo già visto la stringa XML:

```
XmlSerializer Ser= new XmlSerializer(typeof(rssClass));
System.IO.StringWriter SW= new System.IO.StringWriter();
Ser.Serialize(SW,RSS);
string t=SW.ToString();
```

volendo servire la stringa direttamente al richiedente, potremmo fare così:

```
Response.Clear ();
Response.ContentType="xml";
Response.Write (t);
Response.Flush();
```

Se mettessimo il codice appena visto nell'evento onload di una pagina aspx, i nostri utenti potrebbero ricevere il feed semplicemente chiamando tale pagina, per esempio da un link HTML presente in un'altra pagina:

```
<A HREF="paginaRSS.aspx">RSS</A>
```

se invece avessimo scritto il documento in un file XML, avremmo potuto inserire un'ancora al file statico del tipo:

```
<A HREF="paginaRSS.xml">RSS</A>
```

va aggiunto che è nostro dovere controllare che le informazioni immesse siano corrette come formato. Per esempio dovremmo accertarci che le description non contengano HTML, e che le date siano nel formato corretto.
Per esempio se *DateSubmitted* è una variabile di tipo data, potremmo scrivere così:

```
RSS.channel.item[i].pubDate= DateSubmitted.ToString("r");
```

Infatti il parametro di formattazione *"r"* di *ToString()* per una data produce un uscita del tipo *"Thu, 17 Aug 2000 23:32:32 GMT"*, compatibile con il formato voluto da RSS.

E ORA, AL CONTRARIO!

Una cosa interessante è che, una volta ottenute in questo modo, le classi che rappresentano il feed possono essere utilizzate per "deserializzare" un feed XML! Ovvero, è possibile fare l'operazione inversa a quella fatta prima, ottenendo una gerarchia di oggetti valorizzati che rappresentano il feed e che possiamo manipolare agevolmente secondo le comode tecniche OOP. Un abbozzo del codice necessario per fare questo è il seguente, in cui viene utilizzata la classe rss con un feed RSS (in questo caso: *http://dotnet.innovactive.it/rss.aspx*) direttamente ricavato dal Web:

```
XmlSerializer Ser= new XmlSerializer(typeof(rssClass));
System.Net.HttpWebRequest WR=
(System.Net.HttpWebRequest)System.Net.WebRequest.
        Create("http://dotnet.innovactive.it/rss.aspx");
System.Net.WebResponse Resp= WR.GetResponse ();
RSS20.rssClass Rss= new rssClass();
Rss= (RSS20.rssClass)Ser.Deserialize(WR.GetResponse
                                  ().GetResponseStream());
```

CONCLUSIONI

Abbiamo visto come generare una serie di classi per realizzare in modo OOP dei file XML a partire dai loro schemi, poi applicato questa tecnica per ottenere e pubblicare un file RSS.
Il prossimo mese vedremo come salvare un file Word nel nuovo formato WordML, come fare il parsing di questo file per estrarne le caratteristiche salienti, e poi metteremo a frutto il lavoro fatto questo mese per la produzione di un feed RSS che contenga i dati consuntivi dei file Word salvati in una specifica cartella.
Buone sperimentazioni!

Marco Poponi

L'AUTORE

Marco Poponi è laureato in Ingegneria Elettronica. Insegna *Lab di Programmazione* alla facoltà di Ingegneria dell'Università degli Studi di Perugia. È tra i fondatori di *InnovActive Engineering* dotnet.innovactive.it **azienda di sviluppo software e consulenza specializzata nella tecnologia .NET. Si occupa di progettazione, sviluppo e formazione.**

Parsing del Web come fonte di informazioni

Stock Spy La borsa online

Stock Spy è un'applicazione 100% Java che, connettendosi ad Internet, preleva le quotazioni di borsa. In questa nuova serie di articoli vedremo come realizzarla ed esamineremo diversi casi d'utilizzo

Per chi come me è già sotto di qualche migliaio di euro in borsa, e non ha ancora perso il vizio, potrebbe risultare veramente utile un'applicazione che prelevi da Internet le quotazioni in tempo reale ed effettui elaborazioni e statistiche varie. Su Internet si trovano centinaia di siti dove è possibile vedere in tempo reale, quotazioni, grafici, analisi tecniche e statistiche. Addirittura, in alcuni casi, si ha anche l'opportunità di creare un portafoglio personalizzato di titoli. Nonostante questo, scrivere un componente che acceda direttamente alle quotazioni potrà essere utile per due motivi. Il primo è strettamente didattico: impareremo a realizzare una connessione con un server Http ed eseguire il parsing dei risultati. Il secondo è di natura più pratica: avendo a disposizione le quotazioni sarà possibile implementare un'applicazione che le elabori in qualsiasi modo riteniate opportuno. Realizzando moduli fatti ad hoc, potrete seguire l'andamento di un'azione e di conseguenza rendervi conto se è il caso di investire o meno. Oppure creare i vostri tool di statistiche e analisi tecnica, così finalmente potrete perdere i vostri soldi grazie ad un programma scritto da voi e non da altri, il che è molto confortante.

L'ARCHITETTURA

L'idea base di Stock Spy è quella di "spiare" le quotazioni dei titoli di borsa in tempo reale. Il nucleo dell'applicazione consiste in un modulo che si connette, mediante Http, ad un provider di dati. Chi è il provider di dati? Beh, uno dei tanti siti web che offre le quotazioni di borsa in tempo reale. In definitiva, la nostra applicazione simulerà un browser che accede ad una determinata URL, ricevendo in output una pagina html contenente le quotazioni di borsa. In Fig. 1 è mostrata l'architettura del nucleo di Stock Spy. Esso è composto da un modulo chiamato *HttpConnector* e da un altro denominato parser. L'*HttpConnector* accede ad un sito Internet (specificato da una URL) inoltrando una richiesta Http. Questa URL è la stessa che, se specificata in un browser, visualizzerebbe la lista delle quotazioni delle azioni. La risposta sarà una pagina html contenente le quotazioni. A questo punto entra in gioco il parser che ricercherà nel codice html le azioni, le relative quotazioni ed altre informazioni utili. Da quanto detto emerge che il corretto funzionamento di Stock Spy dipende dal provider dei dati, cioè dal sito scelto per prelevare le quotazioni. Infatti sia l'URL e sia il parser saranno diversi a seconda del sito target. Inoltre, se uno stesso sito in futuro deciderà di rappresentare in modo diverso la risposta, anche Stock Spy dovrà adeguarsi di conseguenza. Le limitazioni esistono, ma qui stiamo parlando di una "Spy", quindi un minimo di versatilità è comunque richiesto. Questo aspetto sarà dovutamente preso in considerazione; infatti implementeremo il connettore ed il parser in modo che sia possibile, quando richiesto, cambiare l'implementazione a run-time senza modificare il framework dell'applicazione.

CONNESSIONE HTTP

Implementeremo il modulo *HttpConnector*, visto nell'architettura introdotta nel paragrafo precedente, mediante la classe Java *HttpConnector*. Questa classe userà il package *java.net* ed in particolare la classe *HttpURLConnection* per instaurare una con-

NOTA

Poco prima che questo articolo fosse mandato in stampa, Katabeweb finanza ha modificato il proprio sito, cambiando sia le URL sia il codice HTML. Purtoppo il CD allegato alla rivista era già stato stampato; per questo motivo i listati che troverete lì non sono aggiornati. Tuttavia, è possibile scaricare i listati aggiornati dal sito ufficiale di ioProgrammo.

REQUISITI

Conoscenze richieste
Elementi di Java

Software
J2SE 1.4.1 o superiore

Impegno

Tempo di realizzazione

nessione tra la nostra applicazione ed un server Http. Il costruttore della classe prenderà in input una stringa rappresentate una URL e la memorizzerà nella variabile locale *url*:

```
import java.io.*;
import java.util.*;
import java.net.*;
public class HttpConnector {
  private String url = null;
  public HttpConnector(String url) {this.url = url; }
...
```

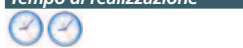
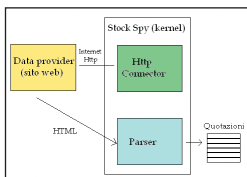



Fig. 1: **L'architettura del nucleo (kernel) di Stock Spy**

Per effettuare una richiesta Http è necessario inviare un comando *GET* o *POST* al server. Il metodo *get* si occuperà di tale compito:

```
public String get() throws Exception {
  URL u = new URL(url);
  HttpURLConnection huc =
   (HttpURLConnection) u.openConnection();
  huc.setRequestMethod("GET");
  huc.setDoOutput(true);
  huc.connect();
  huc.getResponseCode();
  return createResponse(huc); }
```

Il metodo crea una nuova connessione (*huc*) con la URL specificata da *url*. Successivamente viene impostato il tipo di richiesta *(GET)* e il flag *DoOutput* a *true* (significa che l'output è richiesto). A questo punto avviene la connessione, mediante l'invocazione del metodo *connect*. Il metodo *createResponse* avrà il compito di prelevare la risposta e restituirla come stringa. Ecco l'implementazione:

```
private String createResponse(HttpURLConnection huc)
  throws Exception {
  ByteArrayOutputStream out =
    new ByteArrayOutputStream();
  InputStream in = huc.getInputStream();
  int i=0;
  while ((i = in.read())!=-1) {
    out.write(i);}
  in.close();
  out.close();
  huc.disconnect();
  return new String(out.toByteArray()); }
```

LISTATI

Nel CD allegato alla rivista troverete i listati completi delle classi introdotte nell'articolo. Inoltre sarà presente la classe *StockSpy* che contiene esempi d'invocazione di tutti i metodi presenti nell'interfaccia StockConnector.

Il metodo accede all'input stream rappresentante la risposta, legge i byte e li scrive in un *ByteArrayOutputStream* che, alla fine, sarà utilizzato per creare la stringa da restituire. Siccome per alcuni siti è necessario specificare un comando *POST* anziché *GET,* la classe *HttpConnector* implementerà anche il metodo *post:*

```
public String post(String postString)
  throws Exception {
  URL u = new URL(url);
  HttpURLConnection huc = (HttpURLConnection)
                                    u.openConnection();
  huc.setRequestMethod("POST");
  huc.setDoOutput(true);
  huc.connect();
  OutputStream os = huc.getOutputStream();
  os.write(postString.getBytes());
  int code = huc.getResponseCode();
  return createResponse(huc); }
```

Come si può notare, il metodo *post* è molto simile al metodo *get,* eccetto per due piccole ma sostanziali differenze: il metodo impostato per l'oggetto *HttpURLConnection* adesso è *"POST"* e la stringa rappresentate le informazioni da "postare", ovvero *postString,* sarà scritta nell'*OutputStream* dell'oggetto *huc* subito dopo la connessione. La classe *HttpConnector* potrà essere utilizzata per effettuare una connessione a qualsiasi URL, utilizzando sia il comando *GET* sia il comando *POST.* Questa classe ci servirà per effettuare una richiesta Http al provider dei dati, cioè al sito scelto per ottenere le quotazioni di borsa.

STOCK CONNECTOR

Sebbene pensata per essere utilizzata in *Stock Spy*, la classe *HttpConnector* è del tutto generica e può instaurare una connessione con qualsiasi URL di un server Http. Il passo successivo è quello di implementare un modulo specifico per le quotazioni che, attraverso *HttpConnector*, effettui la connessione con il provider dei dati.
La classe in questione implementerà la seguente interfaccia:

```
import java.util.*;
public interface StockConnector {
  public void connect(Properties prop)
    throws Exception;
  public StockQuote getQuote(String stockName)
...
    throws Exception;}
```

In pratica, un'implementazione di *StockConnector* sarà in grado di instaurare una connessione con il provider e reperire le quotazioni mediante uno dei metodi dell'interfaccia (vedi Tab. 1 per i dettagli).
Ognuno dei metodi restituisce un array di elementi appartenenti alla classe *StockQuote* così definita:

```
public class StockQuote {
  public String name;
  public double value;
  public Calendar date;
  public double difference; }
```

Ogni quotazione è quindi formata dal nome dell'azione *(name)*, il valore *(value)*, la data dell'ultimo prezzo *(date)* e la variazione in percentuale dal prezzo di riferimento del giorno prima *(difference)*.
Come si può intuire, l'implementazione di *StockConnector* dipende dal provider. A questo punto, per poter continuare, è necessario scegliere uno dei siti web che forniscono le quotazioni di borsa. La mia scelta è caduta su *Kataweb Finanza* poiché fornisce quotazioni *real-time* gratuite ed è facilmente accessibile attraverso un comando *GET.* La dichiarazione della classe *KatawebConnector* che implementa *StockConnector* e l'implementazione del metodo *connect* sono le seguenti:

```
public class KatawebConnector
  implements StockConnector {
  private static String host; //kataweb.host
  private static String mib30; //kataweb.mib30
  private static String mibtel; //kataweb.mibtel
  private KatawebParser parser
    = new KatawebParser();
  public void connect(Properties prop) throws Exception {
     ...
```

La classe ha tre dati membro statici:

host - il nome dell'host http
mib30 - la directory per le quotazioni del mib30
mibtel - la directory per le quotazioni del mibtel

Per rendere l'applicazione flessibile, tali valori saranno prelevati da un file di proprietà:

kataweb.host=http://www.kwfinanza.kataweb.it
kataweb.mib30=/live/azioni/listbluechips.shtml
kataweb.mibtel=/live/azioni/listino/@letter@.shtml

La proprietà *kataweb.mibtel* è parametrizzata rispetto alla lettera con la quale inizia il nome dell'azione; a runtime il riferimento *@letter@* verrà sostituito con la lettera opportuna. Per esempio, se si vuol ottenere la quotazione del titolo *ENI,* l'URL sarà:

http://www.kwfinanza.kataweb.it/live/azioni/listino/e.shtml

Il metodo *connect,* a dispetto del nome, non effettua nessuna connessione. Esso preleva i valori dalle proprietà e li assegna alle tre stringhe statiche. Il motivo del nome connect va ricercato nel fatto che per alcuni siti è necessario autenticarsi prima di accedere al servizio; il codice per l'autenticazione, in questi casi, va inserito proprio in questo metodo. Per il sito *Kataweb Finanza,* l'autenticazione non è necessaria. Come si può notare, la classe ha anche un riferimento ad un'istanza della classe *KatawebParser.* Questa classe è l'implementazione del parser introdotto nell'architettura di Fig. 1 ed ha la responsabilità di prelevare dal codice html le quotazioni e le informazioni relative alle azioni, ed inserirle in oggetti *StockQuote.* Vedremo nel prossimo paragrafo l'implementazione di *KatawebParser* nei dettagli. Avendo a disposizione la classe *HttpConnector,* l'URL del provider e il parser, l'implementazione del metodo *getMib30* diventa una pura formalità:

```
public StockQuote[] getMib30()
  throws Exception {
  HttpConnector connector =
                       new HttpConnector(host + mib30);
  String response = connector.get();
  return parser.parse(response);}
```

Metodo	Scopo
getMib30	Restituisce le quotazioni relative alle azioni appartenenti al Mib 30 (bluechips)
getByLetter	Restituisce le quotazioni relative alle azioni che iniziano per una data lettera
getAllQuotes	Restituisce le quotazioni di tutte le azioni del Mibtel
getCustom	Restituisce le quotazioni relative alle azioni specificate dall'utente
getQuote	Restituisce la quotazione di una singola azione

Tabella 1: I principali metodi per commutare le quotazioni

Il metodo effettua una connessione Http attraverso l'oggetto *connector;* l'URL sarà dato da *host+ mib30,* ovvero *http://www.kwfinanza.kataweb.it/live/azioni/listbluechips.shtml.* La risposta *(response)* verrà elaborata dal parser che restituirà un array di elementi *StockQuote,* contenenti le quotazioni dei titoli del Mib 30. Il metodo *getByLetter* è leggermente più complicato:

```
public StockQuote[] getByLetter(char letter)
  throws Exception {
  String letterPage=mibtel.replaceAll("@letter@",""+letter);
  HttpConnector connector =
    new HttpConnector(host + letterPage);
  String response = null;
  try { response = connector.get();
  } catch (IOException e) { e.printStackTrace();
    return new StockQuote[0]; }
  return parser.parse(response); }
```

Come si può notare, l'URL è ottenuta da *host + letterPage,* dove *letterPage* è la variabile mibtel alla quale la stringa "@letter@" è stata sostituita dalla lettera in input al metodo. Tutti gli altri metodi dell'interfaccia *StockConnector* possono essere implementati a partire da *getByLetter.* L'implementazione non sarà riportata, ma è presente nel codice allegato alla rivista.

PROXY

Se eseguite Stock Spy all'interno di una rete con proxy, sarà necessario specificare le proprietà di sistema *proxySet, proxyHost* e *proxyPort.* L'impostazione può essere effettuata specificano l'opzione *-D* della Java Virtual Machine oppure inserendo le proprietà nel file *stockspy.properties*, come mostrato di seguito:

```
proxySet=true
proxyHost=myproxy.com
proxyPort=3128
```

dove *myproxy.com* sarà il nome del vostro proxy e 3128 la porta.

IL PARSER

Anche il parser, come lo *StockConnector,* dipende

dal provider. Infatti, come abbiamo già detto, l'host http restituisce la risposta in formato html e il modo in cui la pagina è creata è ad assoluta discrezione del web designer del sito. Sta quindi alla nostra applicazione individuare le quotazioni all'interno della pagina html. Questo è il compito del parser, che a livello astratto può essere visto come un'interfaccia:

```
public interface StockParser {
  public StockQuote[] parse(String rawData)
    throws Exception;}
```

L'interfaccia *StockParser* contiene solo il metodo *parse* che prende in input la pagina html *(rawData)* e restituisce un array di quotazioni (*StockQuote*) costruito a partire dai dati individuati nella pagina. Implementiamo quest'interfaccia partendo dalla pagina html restituita dal sito *Kataweb Finanza*. Ecco un esempio della sezione alla quale siamo interessati:

```
<a href="../azioni/scheda/scheda_1683373.shtml">
                                         AUTOSTRADE</a>
</b></td>
<td nowrap align="center" ><b><font
          color="#CC0000">-</font></b></td>
<td align="right">15,640</td>
<td align="right">15:55</td>
<td align="right">-0.19</td>
```

Qui sono presenti tutte le informazioni necessarie: il nome dell'azione, la quotazione, l'ora e la variazione dal giorno precedente. Il parser non dovrà far altro che individuare tali sezioni, per ogni azione, estrarre i risultati ed inserirli in un oggetto *StockQuote*. Per far questo è necessario ricercare all'interno della pagina la stringa: *scheda_1683373.shtml"*> che decreta l'inizio della quotazione di un'azione. Il numero che segue la stringa *"scheda_"* dipende dall'azione, quindi in generale, esprimendoci attraverso le espressioni regolari, dobbiamo individuare l'espressione: *scheda_(\\w*).shtml"*>
A questo punto il gioco è fatto: il nome dell'azione viene immediatamente dopo l'espressione regolare e i tre valori che seguono i tag *<td align="right">* sono rispettivamente la quotazione, l'orario e la variazione. La classe *KatawebParser* procederà nel modo descritto, facendo uso del package per le espressioni regolari presente in Java 1.4.x:

```
import java.util.regex.*;
import java.util.*;
public class KatawebParser
  implements StockParser {
  public StockQuote[] parse(String mainData)
  throws Exception {
    Pattern mainPattern =
          Pattern.compile("scheda_(\\w*).shtml\">");
    Matcher mainMatcher = mainPattern.matcher(mainData);
    ArrayList quotes = new ArrayList();
    while(mainMatcher.find()) {
      String subData = mainData.substring(
                                    mainMatcher.end());
      String name = subData.substring(
                              0,subData.indexOf("<"));
      Pattern stockPattern = Pattern.compile(
                               "<td align=\"right\">");
      Matcher stockMatcher=stockPattern.matcher(subData);
      String info[] = new String[3];
      for(int i=0; i<3; i++) {
        stockMatcher.find();
        String stockData = subData.substring(
                                    stockMatcher.end());
        info[i] = stockData.substring(
                              0,stockData.indexOf("<"));
        info[i] = info[i].replace(',','.');}
      StockQuote stockQuote = new StockQuote();
      stockQuote.name = name;
      stockQuote.value = parseDouble(info[0]);
      stockQuote.date = extractDate(info[1]);
      stockQuote.difference = parseDouble(info[2]);
      quotes.add(stockQuote); }
    StockQuote stockQuote[] = new
                              StockQuote[quotes.size()];
    for (int i = 0; i < stockQuote.length; i++) {
      stockQuote[i] = (StockQuote) quotes.get(i); }
    return stockQuote; }
  private GregorianCalendar extractDate(String raw) {
    GregorianCalendar date = new GregorianCalendar();
    String h[] = raw.split(":");
    if (h.length!=2) {
      h = new String[2]; h[0] = "0"; h[1] = "0"; }
    date.set(date.HOUR_OF_DAY,parseInt(h[0]));
    date.set(date.MINUTE,parseInt(h[1]));
    return date; }
  private double parseDouble(String sdouble) {
    try {
      return Double.parseDouble(sdouble);
    } catch (NumberFormatException e) {
      return 0.0;} }
  private int parseInt(String sint) {
    try {
      return Integer.parseInt(sint);
    } catch (NumberFormatException e) {
      return 0; } }}
```

CONCLUSIONI

In questa prima parte abbiamo gettato le basi per prelevare le quotazioni in tempo reale da Internet ed estrarre le informazioni effettuando il parsing della pagina html in risposta. Nel prossimo articolo vedremo esempi pratici d'utilizzo per la classe *StockConnector.*

Giuseppe Naccarato

SUL WEB

Kataweb Finanza, www.kwfinanza.kataweb.it

G. Naccarato [sito web] www.giuseppe-naccarato.com

• ESPRESSIONI REGOLARI IN JAVA 1.4 ioProgrammo n. 65, *G. Naccarato* (Edizioni Master) gennaio 2003

Funzioni di base e interfaccia grafica

Java, persistenza e fantacalcio

parte seconda

Riprendendo quanto sviluppato nel precedente articolo completiamo l’applicazione di gestione di un torneo di FantaCalcio e ne disegniamo l’interfaccia grafica

Riprendiamo quanto iniziato nel precedente articolo relativamente alla progettazione ed allo sviluppo di un’applicazione Java per la gestione di un torneo di FantaCalcio. Dopo aver illustrato gli strumenti scelti, quali HSqlDb per il database ed OJB quale tool di mapping classi-tabelle, ci proponiamo di mostrare l’implementazione di alcune operazioni di base. Successivo passo avanti è rappresentato dal disegno di un’interfaccia utente che agevoli l’utilizzo dell’applicazione.

Fig. 1: **All’avvio dell’applicazione una splash window accoglie l’utente, mentre in una toolbar sono presenti i pulsanti relativi alle operazioni disponibili**

Si tratta di una semplice applicazione stand-alone pensata per essere utilizzata dalla persona incaricata di svolgere i compiti d’amministrazione del torneo di FantaCalcio.

ARCHITETTURA

Nonostante la semplicità dell’applicazione, è utile fissarne una struttura principale sin dall’inizio, in modo da aver ben chiaro il ruolo dei differenti componenti presenti. Osservando la Fig. 2, si può notare che è presente uno strato inferiore relativo alle classi di business. Esse rappresentano le entità (*Squadra*, *Calciatore*, ecc.) su cui si basano le funzionalità dell’applicazione. Tali classi sono state già

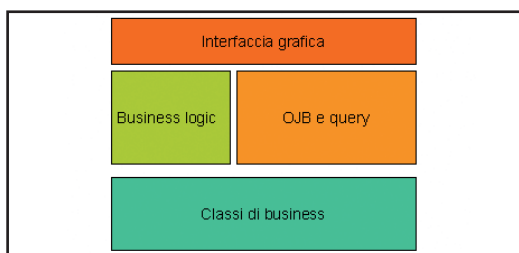



Fig. 2: **È consigliabile separare sempre la logica di business da quella di presentazione dei dati**

progettate nel precedente articolo ed organizzate in un package denominato *fcalcio.model*. La maggior parte di tali classi è persistente sul database HSqlDb ed a tale scopo è necessaria la presenza di un componente apposito (descritto in Fig. 2 come *“OJB e query”*). In esso sono presenti le classi che contengono la logica di persistenza, ossia conoscono come utilizzare (in questo caso) l’interfaccia ODMG per inserire, aggiornare e cercare oggetti di business all’interno del database. Al fianco di tali classi è presente un componente cosiddetto di business logic, il cui compito è utilizzare opportunamente i mattoncini messi a disposizione dalle classi di business per costruire delle funzionalità più complesse (ad esempio la creazione di un calendario). Tale componente può utilizzare le operazioni di base fornite dallo strato di persistenza. Infine al livello più elevato si trova il componente comprendente le classi relative all’interfaccia grafica. Qui si trova la logica di presentazione dei dati che potrebbe essere modificata indipendentemente dal business.

BUSINESS LOGIC E PERSISTENZA

Procediamo col definire innanzitutto le funzionalità del core dell’applicazione, presenti nello strato della

business logic e della persistenza. Tali classi sono organizzate in un package distinto, denominato *fcalcio.engine*. Il class diagram di tale package è illustrato in Fig. 3.
Una classe d’utilità particolarmente importante è *OJBConnection*, un singleton, il quale offre dei metodi che permettono di gestire le connessioni di OJB verso il database. Ricordiamo che l’interfaccia Java che utilizziamo verso OJB per la gestione della persistenza è quella definita come ODMG. Il metodo principale, *getDatabase*, restituisce un’istanza del database a cui è connesso. Notare che la prima volta la connessione viene creata ed aperta utilizzando quale alias del db il valore *(“fantaCalcio”)* contenuto nella configurazione di OJB, *repository_ database*.

Fig. 3: *Tra le principali classi responsabili dell’implementazione delle funzionalità dell’applicazione sono presenti alcune dedicate alle query*

```
private static final String DB_ALIAS = "fantaCalcio";
    private Database db = null;
    private Implementation odmg = null;
    ...
    public Database getDatabase() throws ODMGException {
        if(db != null)
            if(((DatabaseImpl) db).isOpen())
                    return db;
            odmg = OJB.getInstance();
            db = odmg.newDatabase();
            db.open(DB_ALIAS,
                        Database.OPEN_READ_WRITE);
            return db; }
```

Un altro metodo messo a disposizione da *OJBConnection* è quello relativo alla creazione di una transazione. Esso utilizza il riferimento alla Implementation utilizzata, in questo caso ODMG.

```
    public Transaction createTransaction() throws
                                    ODMGException {
            Transaction tx = odmg.newTransaction();
            tx.begin();
            return tx; }
```

Da un esame del package *fcalcio.engine*, possiamo notare che sono presenti anche altre classi quali *CampionatoManager, SquadraManager*, ecc. Esse sono classi, utilizzate all’interno delle funzionalità di base, che offrono metodi statici d’esecuzione di query su classi diverse. Ad esempio *SquadraManager* contiene tutte le possibili query da eseguirsi sulla classe *Squadra*.

```
public static Squadra trovaSquadra(String nome) {
  PersistenceBroker broker = null;
  try {
    broker = PersistenceBrokerFactory.
                        defaultPersistenceBroker();
    Criteria crit = new Criteria();
    crit.addEqualTo("nome", nome);
    Query query = new QueryByCriteria(Squadra.class, crit);
    java.util.Iterator iter = broker.getCollectionByQuery(
                                query).iterator();
    if(iter.hasNext())
        return (Squadra) iter.next();
    return null;
  } finally {
    if(broker != null)
      broker.close(); } }
```

La prima parte del codice si occupa di reperire un *PersistentBroker* che rappresenta il punto di accesso più vicino al database. Notare quindi che la clausola *where* della query viene definita creando un criterio, in cui si utilizza l’attributo della classe (in questo caso *nome*) e non la colonna della tabella. In seguito si crea un oggetto *Query* indicando la classe su cui effettuare la ricerca. Il risultato consiste in oggetti *Squadra* che soddisfano i criteri indicati. Le principali funzionalità di business, individuate nel precedente articolo, sono messe a disposizione dai metodi di un’interfaccia, denominata *FCModel*, del tipo

```
public String[] getSquadre() throws BusinessException;
public Campionato creaCampionato(String stagione,
 String[] nomiSquadre,Date da) throws BusinessException;
public void aggiornaVoti(String fileName) throws
                                BusinessException;
```

Il motivo è che in tal modo si crea un disaccoppiamento tra chi utilizza tali metodi (ad esempio lo strato dell’interfaccia grafica) e l’implementazione degli stessi. La classe d’implementazione invece è rappresentata da *FCModelImpl*. Il suo costruttore apre una connessione al database utilizzando la classe di utilità *OJBConnection*.

```
public FCModelImpl() throws ODMGException {
    OJBConnection.getInstance().getDatabase(); }
  public void close() throws ODMGException {
```

QUERY NEL MONDO OO

A differenza del modello relazionale, nel mondo OO il reperimento degli oggetti sul database dovrebbe avvenire di regola navigando sulle associazioni. A tal fine è necessario un buon design delle classi così da rendere l’applicazione molto più veloce. In questo caso le query si riducono spesso a semplici lookup, a parte le classiche funzionalità di report, aggregazione, ecc.

```
    Database db = OJBConnection.getInstance(
                                    ).getDatabase();
      db.close();
}
```

Tra le funzionalità di base, osserviamo il metodo che fornisce la creazione di un nuovo campionato. Il metodo *creaCampionato* riceve una stringa che identifica la stagione (ad esempio *"2003-04"*), un array di sei nomi di squadre presenti nel database, ed una data di partenza da cui generare il calendario.

```
Transaction tx = OJBConnection.getInstance(
                              ).createTransaction();
// recupera le squadre
Squadra squadre[] = new Squadra[nomiSquadre.length];
for(int k=0;k<nomiSquadre.length;k++) {
squadre[k] = SquadraManager.trovaSquadra(
                                  nomiSquadre[k]);
  ... }
Campionato campionato = new Campionato();
  campionato.setStagione(stagione);
  ...
  tx.lock(campionato, tx.WRITE);
  tx.commit();
```

Possiamo osservare che la creazione di un nuovo oggetto *Campionato* avviene mediante il solito costrutto Java *new*. In seguito, dopo aver posto un lock in scrittura sull'oggetto (ed eseguito il commit della transazione), l'oggetto è salvato sul database.
Notare che al momento della creazione di un oggetto *Campionato* vengono istanziati anche una serie di altri oggetti *Giornata* e *Partita* che rappresentano il calendario. Per essi non è necessario porre esplicitamente nessun lock in quanto OJB, alla creazione di una nuova istanza, propaga la persistenza a tutti gli oggetti raggiungibili dall'oggetto su cui si è messo il lock *(cascade)*. Un'altra funzionalità da esaminare è quella che crea una formazione per una squadra specificata.

```
public Formazione creaFormazione(Squadra squadra)
                        throws BusinessException {
  try {
    Transaction tx = OJBConnection.getInstance(
                              ).createTransaction();
    Formazione f = new Formazione();
    tx.lock(squadra,tx.WRITE);
    squadra.setFormazione(f);
    tx.commit();
    ...
```

In tale situazione, occorre creare un nuovo oggetto (della classe *Formazione*) e modificarne un altro (della classe *Squadra*), dato che esiste un'associazione tra i due. Si pone dapprima un lock sull'oggetto *Squadra* da modificare ed in seguito s'imposta l'associazione con la nuova istanza *Formazione*.
Passiamo ora all'implementazione dell'aggiornamento dei voti; ricordiamo che tale fase è necessaria prima del calcolo dei risultati, in quanto occorre, per ogni calciatore, reperire e valorizzare nella classe *UltimaPartita,* i giudizi (voto in pagella, numero di goal segnati, ammonizioni, ecc.) riportati sul giornale di riferimento. In questo caso immaginiamo di fornire all'applicazione un file di testo, denominato *voti.txt,* dove sono riportate tali informazioni in un semplice formato consistente di una serie di campi separati da tabulazioni.

```
# Cognome nome voto golf gols rigp rigs autog amm esp
ABBIATI Christian  60  0   1   0   0    0    0   0
```

Il metodo di *business,* ricevendo come parametro il nome del file da leggere, lo apre e per ogni riga letta ne estrae i dati. In base al nome ed al cognome recupera, mediante la classe *CalciatoreManager,* l'oggetto calciatore referenziato. Si crea un'istanza di *UltimaPartita* nel caso questa non esista, si pone un lock sull'oggetto *Calciatore* e s'imposta l'associazione. Viceversa in caso in cui un oggetto *UltimaPartita* già è associato, poiché occorre modificare solo questo, è sufficiente porre un lock solo su tale istanza.

```
public void aggiornaVoti(String fileName) throws
                              BusinessException {
  ...
  line = in.readLine();
  ...
  Calciatore c = (Calciatore) list.get(0);
  UltimaPartita partita = c.getUltimaPartita();
  if(partita == null) {
    partita = new UltimaPartita();
    tx.lock(c,tx.WRITE);
    partita.setAmmonizioni(Integer.parseInt(amm));
    ...
    c.setUltimaPartita(partita); }
  else {
    tx.lock(partita,tx.WRITE);
    partita.setAmmonizioni(Integer.parseInt(amm));
    ... }
```

Le implementazioni dei restanti metodi non differiscono di molto da quanto visto e pertanto possiamo passare ad esaminare l'ultimo strato dell'applicazione, ma non per questo il meno importante.

L'INTERFACCIA GRAFICA

Le classi relative a tali componenti sono organizzate all'interno del package *fcalcio.gui.* Il punto di partenza dell'applicazione è dato dalla classe *FCal-*

PERSISTENCEBROKER
Rappresenta il motore, a livello più basso di OJB, che gestisce il mapping O/R, il recupero di oggetti mediante semplici query, lo storing e la cancellazione di oggetti. Ogni interfaccia, ODMG o JDO, è basata su questo strato.

cioFrame che contiene la toolbar in cui sono presenti i diversi pulsanti che eseguono le possibili operazioni. Il coordinatore di tutti gli eventi grafici relativi alle funzionalità dell'applicazione è la classe *FCalcioMediator.* Essa implementa un'interfaccia, *FCalcioEventListener,* che fornisce i metodi relativi alle operazioni richiamate cliccando sui pulsanti della toolbar, come ad esempio:

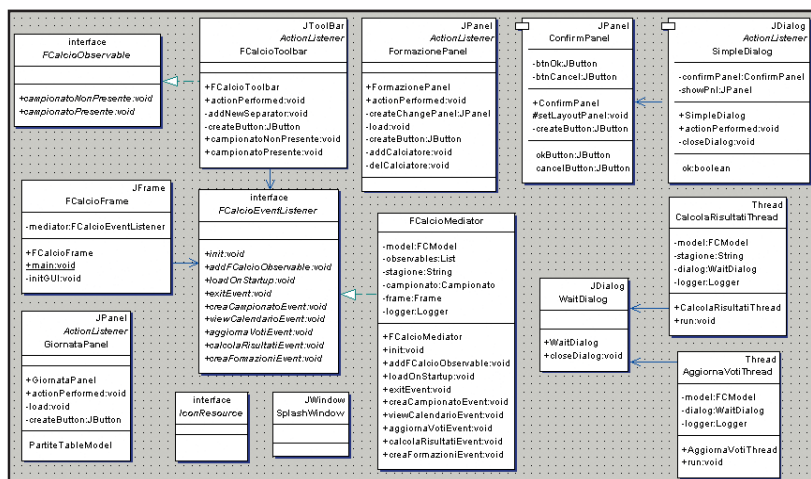



Fig. 4: ***Il cuore dell'interfaccia grafica è rappresentato dal pattern Mediator***

```
void exitEvent();
    void creaCampionatoEvent();
    void viewCalendarioEvent();
    void aggiornaVotiEvent();
    void calcolaRisultatiEvent();
    void creaFormazioniEvent();
```

Analizzeremo l'implementazione più avanti; per ora esaminiamo la classe *FCalcioToolbar,* derivante da JToolBar, che contiene gli oggetti JButton dei pulsanti ed implementa l'interfaccia *ActionListener.* Ogni volta che si clicca su un pulsante, viene richiamato il metodo di gestione dell'evento che a sua volta delega l'operazione sul corretto metodo dell'interfaccia *FCalcioEventListener* registrata.

```
public void actionPerformed(ActionEvent e) {
   if(listener == null)
   return;
   if(e.getSource().equals(btnCreaCampionato)) {
      listener.creaCampionatoEvent();
   return; }
```

Un'altra interfaccia utile è *FCalcioObservable.* Essa definisce due soli metodi.

```
    void campionatoNonPresente();
    void campionatoPresente();
```

La loro invocazione permette di segnalare all'entità, che implementa tale interfaccia, l'informazione se è presente o meno nel database un campionato. Tale interfaccia è implementata proprio dalla classe *FCalcioToolBar* che utilizza tale informazione per abilitare o disabilitare alcuni pulsanti in base allo stato dei dati. Ad esempio se un campionato non è presente, viene attivato il solo pulsante relativo alla creazione di un nuovo campionato.

```
public void campionatoNonPresente() {
        btnCreaCampionato.setEnabled(true);
        btnViewCalendario.setEnabled(false);
        btnAggiornaVoti.setEnabled(false);
        btnCalcolaRisultati.setEnabled(false);
        btnCreaFormazioni.setEnabled(false);
}
```

L'interfaccia *FCalcioEventListener* permette di definire, mediante il metodo *addFCalcioObservable,* le entità *FCalcioObservable.* In tal modo è proprio la classe che agisce da mediator ad inviare le suddette informazioni. Diamo ora uno sguardo in maggior dettaglio a tale classe.

```
public class FCalcioMediator implements
                                    FCalcioEventListener {
    private FCModel model = null;
    private List observables = null;
    public FCalcioMediator(String stagione) {
        observables = new ArrayList();
        try {
            this.stagione = stagione;
             model = new FCModelImpl();
        } catch(ODMGException oe)
    {
```

Come possiamo notare, il suo costruttore crea un oggetto *FCModelImpl* che contiene i metodi di business da invocare e riceve una stringa con l'identificativo della stagione da utilizzare (nel nostro esempio *"2003-04"*) che servirà per creare o ricercare il relativo campionato. Viene inoltre preparata una lista in cui mantenere gli oggetti *FCalcioObservable* registrati. All'avvio dell'applicazione il mediator esegue un metodo *loadOnStartup* con cui recupera l'oggetto *Campionato* dal database.

```
public void loadOnStartup() {
   try {
      campionato = model.getCampionato(stagione);
      for(int k=0;k<observables.size();k++) {
         FCalcioObservable obj = (FCalcioObservable)
                                           observables.get(k);
         if(campionato == null)
           obj.campionatoNonPresente();
         else
           obj.campionatoPresente();}
   } catch(BusinessException be) {
      Logger.getLogger("fcalcio").error(be.getMessage());
      exitEvent();} }
```

LOG4J

È una libreria della Apache Software Foundation che offre un sistema di logging utile per qualsiasi applicazione. Basta agire su di un file di configurazione (log4j.properties o log4j.xml) per modificare a run-time la logica del logging.

Notare che a secondo che l'oggetto richiesto viene trovato o meno, si segnala un diverso evento agli oggetti *FCalcioObservable*. Una funzionalità fornita dal mediator è quella di visualizzare il calendario del campionato.

```
public void viewCalendarioEvent() {
   GiornataPanel panel = new GiornataPanel(campionato
      .getGiornate(),campionato.getProssimaGiornata());
   SimpleDialog dialog = new SimpleDialog(
                  frame,"Calendario " + stagione,panel);
   dialog.setVisible(true);
}
```

In tal caso il metodo *viewCalendarioEvent* crea un apposito oggetto *GiornataPanel* (fornendogli la lista

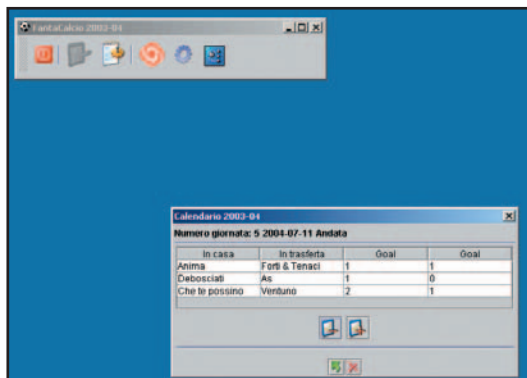



Fig. 5: **All'apertura del pannello viene mostrata automaticamente la prossima giornata da giocare**

delle giornate e l'indicazione della prossima giornata da giocare) e lo inserisce all'interno di una finestra di dialogo. Un'altra funzionalità, leggermente più complessa, è quella d'inserimento e visualizzazione delle formazioni il cui risultato è mostrato in

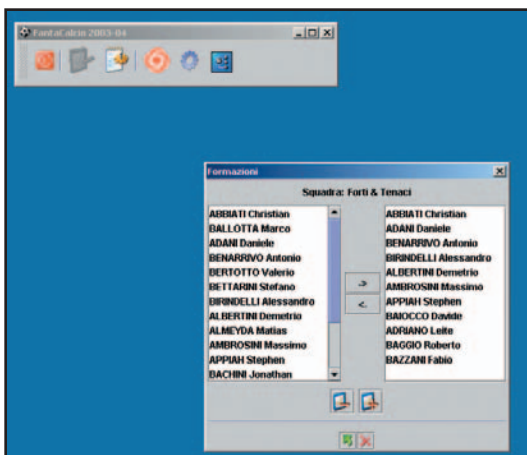



Fig. 6: **Per semplicità non sono effettuati controlli sui possibili moduli utilizzabili**

Fig. 6. Altre operazioni disponibili sono quelle d'aggiornamento dei voti dei calciatori e di calcolo dei risultati. Nel primo caso abbiamo visto che il metodo di business legge un semplice file di testo. Il metodo *aggiornaVotiEventi* del mediator semplicemente visualizza una finestra di dialogo che mostra l'indicazione dell'operazione in corso ed avvia un *thread*, *AggiornaVotiThread*, che invoca il metodo *aggiornaVoti* sul relativo oggetto *FCModel*.

```
public void aggiornaVotiEvent() {
   WaitDialog wait = null;
   wait = new WaitDialog(frame,"Aggiornamento
               voti","Lettura file voti.txt. Attendere...");
   AggiornaVotiThread thread = new
                        AggiornaVotiThread(model,wait);
   thread.setPriority(Thread.MIN_PRIORITY);
   thread.start();
   wait.setVisible(true); }
```

La funzionalità di calcolo dei risultati non si discosta molto da quella d'aggiornamento dei voti. In tale caso viene eseguito il metodo di business *calcolaRisultati* disponibile su *FCModel* con l'effetto di valorizzare i risultati delle partite della giornata in corso.

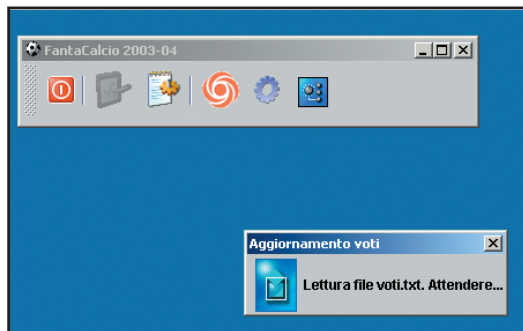



Fig. 7: **A fronte dell'operazione d'aggiornamento, ogni Calciatore ha associato un oggetto UltimaPartita valorizzato con i dati letti**

UTILIZZO E CONFIGURAZIONE

Tutto l'occorrente per compilare ed eseguire l'applicazione è contenuto all'interno di una directory nominata codice. Per compilare l'applicazione è disponibile un file *build.xml* per *Ant*. È sufficiente posizionarsi sulla directory codice e digitare *ant* da una finestra dei comandi. I file compilati verrano generati nella *directory classes*. Per utilizzare l'applicazione occorre posizionarsi nella directory *test*. Essa contiene i file di configurazione di OJB contenenti i metadata definiti nel precedente articolo e la connessione al database. È presente inoltre il file di configurazione di *Log4J*, la libreria utilizzata per il logging, ed una directory log in cui sono destinati i relativi file.
L'esempio fornito con l'articolo offre già un database *HSqlDb*, denominato *FantaCalcioDb* e pronto per essere utilizzato.

David Visicchio

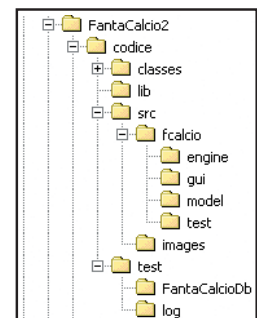



Fig. 8: **Nella directory src sono presenti i file sorgenti mentre nella lib i jar utilizzati**

L'AUTORE

David Visicchio è laureato in Ingegneria. Specializzato nella persistenza e nei sistemi middleware di sistemi object-oriented, i suoi interessi sono orientati principalmente alle architetture distribuite basate su piattaforme J2EE e J2SE.

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

RECUPERARE LA STRUTTURA DELLE DIRECTORY

Semplice funzione per verificare l'organizzazione delle directory all'interno del sistema.

```
'Dichiarazioni
Public Declare Function GetSystemDirectory Lib "kernel32" Alias
        "GetSystemDirectoryA" (ByVal lpBuffer As String, ByVal nSize
                                                 As Long) As Long
'Codice
Public Function mysystemdirectory()
Dim m_mybuf As String * 25
Dim m_val As Long, systemdirectory As String
m_val = GetSystemDirectory(m_mybuf, 25)
systemdirectory = Left(m_mybuf, InStr(m_mybuf, Chr(0)) - 1)
MsgBox systemdirectory
End Function
private sub form_load()
mysystemdirectory
end sub
```

COME ESEGUIRE UN CD AUDIO

Una routine per ascoltare un CD Audio da un'applicazione VB. La dichiarazione va inserita nella sezione generale di un modulo *BAS*.

```
Public Declare Function mciSendString Lib "winmm.dll" Alias _
"mciSendStringA" (ByVal lpstrCommand As String, ByVal _
lpstrReturnString As String, ByVal uReturnLength As Long, _
ByVal hwndCallback As Long) As Long
'creare la seguente routine:
Sub cmdPlay_Click ()
    Dim lRet As Long, nCurrentTrack As Integer
'apertura del device
    lRet = mciSendString("open cdaudio alias cd wait", 0&, 0, 0)
'setta il formato alle Tracce (per default è millisecondi)
    lRet = mciSendString("set cd time format tmsf", 0&, 0, 0)
'suona dall'inizio del CD
    lRet = mciSendString("play cd", 0&, 0, 0)
```

IL TIP DEL MESE
UN PORT-SCAN IN VISUAL BASIC

Questo progetto svolge il compito di un completo *Port Scan*. Dato un indirizzo IP (oppure il nome dell'host) e specificando un range di porte, il programma restituisce tutte le porte che risultano aperte. Sul CD è presente il progetto VB.

Tip fornito dal sig. Rosario Sansale

```
Option Explicit
Private Sub cmdEsci_Click()
Unload Me
End Sub
Private Sub cmdStart_Click()
If txtCampo(0).Text = vbNullString Then
  MsgBox LoadResString(101), vbExclamation, LoadResString(103)
  txtCampo(0).SetFocus
ElseIf txtCampo(1).Text = vbNullString Or txtCampo(2).Text =
                                              vbNullString Then
  MsgBox LoadResString(102), vbExclamation, LoadResString(103)
ElseIf txtCampo(1).Text > txtCampo(2).Text Then
  MsgBox LoadResString(105), vbExclamation, LoadResString(103)
Else
  If cmdStart.Caption = "&Scan" Then
    cmdStart.Caption = "&Stop"
    lstLog.Clear
    Winsock1.RemotePort = txtCampo(1).Text
    InitWinSockServer Winsock1
    Timer1.Enabled = True
  Else
    cmdStart.Caption = "&Scan"
    Timer1.Enabled = False
    Winsock1.Close
  End If
End If
```

```
'o da una traccia specifica, ad es. la 4
'nCurrentTrack = 4
'lRet = mciSendString("play cd from" & Str(nCurrentTrack), 0&, 0, 0)
End Sub
'chiude il device al termine dell'esecuzione
Sub cmdStop_Click ()
        Dim lRet As Long
'arresta la riproduzione audio
        lRet = mciSendString("stop cd wait", 0&, 0, 0)
        DoEvents
'chiude il device
        lRet = mciSendString("close cd", 0&, 0, 0)
End Sub
```

JAVA

LA DIRECTORY IN UNO ZIP

Attraverso il package *java.util.zip*, Java fornisce alcuni semplici meccanismi per comprimere e decomprimere file. Il metodo *zipDir* mostra come comprimere ricorsivamente un albero di directory. Il metodo accetta un oggetto *String* come directory da zippare e un oggetto *ZipOutputStream* che rappresenta il file zippato. Il metodo *zipDir* non include le directory vuote nell'archivio zip creato.

```
try {
    //crea un ZipOutputStream nel quale zippare i dati
    ZipOutputStream zos = new
           ZipOutputStream(new FileOutputStream(".\\curDir.zip"));
    //partiamo dal presupposto che esista un directory chiamata inFolder
    //chiama il metodo zipDir
    zipDir(".\\inFolder", zos);
    //close the stream
    zos.close(); }
catch(Exception e)
{ //cattura le eccezioni}
//codice per il metodo
public void zipDir(String dir2zip, ZipOutputStream zos)  {
    try    {
        //crea un nuovo oggetto File
           zipDir = new File(dir2zip);
        //ottiene l'elenco del contenuto delle directory
        String[] dirList = zipDir.list();
        byte[] readBuffer = new byte[2156];
        int bytesIn = 0;
        //Il loop seguente scansiona ricorsivamente l'albero delle
        //directory comprime i file individuati
        for(int i=0; i<dirList.length; i++)  {
            File f = new File(zipDir, dirList[i]);
        if(f.isDirectory())  {
            //se l'oggetto file è una directory chiama  nuovamente la
            //funzione per aggiungere nell'archivio tutto il contenuto delle
                                                 directory ricorsivamente
            String filePath = f.getPath();
            zipDir(filePath, zos);
                //ripete nuovamente il ciclo
            continue; }
            //se arriva qui, significa che l'oggetto File f
            //non è una directory,
            //quind crea un FileInputStream invece di f
            FileInputStream fis = new FileInputStream(f);
        ZipEntry anEntry = new ZipEntry(f.getPath());
            zos.putNextEntry(anEntry);
            //scrive il contenuto del file in ZipOutputStream
            while((bytesIn = fis.read(readBuffer)) != -1)  {
                zos.write(readBuffer, 0, bytesIn);  }
           //chiude lo Stream
           fis.close();  } }
catch(Exception e)
{  //cattura le eccezioni }
```

COME TRASFERIRE FILE DA UN SERVER A UN CLIENT

Utilizzare le classi *DataInputStream* e *DataOutputStream* e i socket Java per trasferire file da un server a un client. Nell'esempio seguente client e server sono in esecuzione sulla stessa macchina. L'applicazione è composta da due classi.

```
// Client.java
import java.net.*;
import java.io.*;
  public class Client {
    public static void main(String[] args)
    throws IOException {
     InetAddress addr = InetAddress.getByName(null);
        System.out.println("addr = " + addr);
         Socket socket =new Socket("127.0.0.1", 8080);
try {
        System.out.println("socket = " + socket);
     BufferedReader in = new BufferedReader(new
                      InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(new BufferedWriter(new
              OutputStreamWriter(socket.getOutputStream())),true); }
finally {
      System.out.println("Arresto il...");
socket.close(); } }
}
// Server.java
import java.io.*;
import java.net.*;
public class Server{
   public static final int PORT = 8080;
     public static void main(String[] args)
     throws IOException {
        ServerSocket s = new ServerSocket(PORT);
         System.out.println("Avviato: " + s);
try { Socket socket = s.accept();
try { System.out.println("Connessione accettata: "+ socket);
         BufferedReader in =new BufferedReader(new
                      InputStreamReader(socket.getInputStream()));
          PrintWriter out = new PrintWriter(new BufferedWriter(new
               OutputStreamWriter(socket.getOutputStream())),true);}
finally {
```

```
    System.out.println("Arresto...");
socket.close();} }
finally {s.close();} }
}
```

USARE LE IMMAGINI COME ETICHETTE DI PULSANTI AWT

I pulsanti creati con la classe *JButton*, appartenente al package *javax.swing*, oltre al semplice testo, possono contenere anche delle immagini, mentre in genere questo non è permesso con i pulsanti creati con la classe *Button* del package *java.awt*.
Il tip seguente consente di inserire immagini nei pulsanti creati con *AWT*.

```
import java.awt.*;
import java.awt.event.*;
class ImageButton extends Button implements MouseListener {Image i[];
  int select=1;
  int w=0;
  int h=0;
  int iw,ih;
    public ImageButton(Image im[]) {
      super(" ");
  i=new Image[4];
  i=im;
  iw=i[0].getWidth(this);
  ih=i[0].getHeight(this);
        setSize(iw,ih);
          addMouseListener(this);}
      public Dimension getPreferredSize() {
        return new Dimension(iw,ih); }
      public Dimension getMinimumSize() {
        return new Dimension(iw,ih); }
      public void setBounds(int x,int y,int width, int height) {
w=width;
h=height;
        super.setBounds(x,y,width,height);}
      public void setSize(int width,int height) {
w=width;
h=height;
       super.setSize(width,height); }
     public void setEnabled(boolean e) {
        if(e) { select=1;}
        else {
          select=0; }
        repaint();
          super.setEnabled(e); }
     public void paint(Graphics g) {
       g.drawImage(i[select],0,0,w,h,this);}
     public void mouseClicked(MouseEvent e) {}
     public void mouseEntered(MouseEvent e) {
       if(select!=0) {
           select=2;
           repaint(); } }
     public void mouseExited(MouseEvent e) {
       if(select!=0) {
           select=1;
           repaint(); } }
    public void mousePressed(MouseEvent e) {
      if(select!=0) {
           select=3;
           repaint(); }}
    public void mouseReleased(MouseEvent e) {
      if(select!=0) {
           select=2;
           repaint(); } }
}
```

DRAG AND DROP IN JAVA

Il modo più semplice per inserire il supporto drag and drop all'interno di un programma Java, consiste nell'utilizzare il metodo *setDragEnabled()* con argomento *true* sul componente.
L'applicazione seguente mostra come aggiungere il trascinamento di testo nelle proprie applicazioni.

```
import java.awt.*;
  import java.awt.event.*;
  import javax.swing.*;
  import java.text.*;
  import java.util.*;
  public class Drag1 {
    public static void main(String args[]) {
      JFrame frame = new JFrame("Drag");
      frame.setDefaultCloseOperation(
          JFrame.EXIT_ON_CLOSE);
      Container content = frame.getContentPane();
      JPanel inner = new JPanel(new BorderLayout());
      JTextArea top = new JTextArea(5, 40);
      top.setDragEnabled(true);
      JScrollPane topScroll = new JScrollPane(top);
      inner.add(topScroll, BorderLayout.NORTH);
      JTextArea bottom = new JTextArea(5, 40);
      bottom.setDragEnabled(true);
      JScrollPane bottomScroll =
          new JScrollPane(bottom);
      inner.add(bottomScroll, BorderLayout.SOUTH);
      content.add(inner, BorderLayout.CENTER);
      DateFormatSymbols symbols =
          new DateFormatSymbols(Locale.US);
      JList list = new JList(symbols.getWeekdays());
      list.setDragEnabled(true);
      JScrollPane leftScroll = new JScrollPane(list);
      content.add(leftScroll, BorderLayout.WEST);
      frame.pack();
      frame.show(); }
  }
```

C#

CANCELLARE UNA DIRECTORY RICORSIVAMENTE

Mediante l'overload del metodo *Delete* della classe *Directory* e impostando il secondo argomento di tale metodo a *true*, è possi-

bile cancellare il contenuto di una directory in modo ricorsivo.

```
using System.IO;
class TestDirectory {
public static void Main() {
Directory.Delete("C:\\testc_sharp", true);}
}
```

LEGGERE IL CONTENUTO DI UN FILE DI TESTO

La funzione seguente riceve come parametro il percorso del file di testo da leggere e restituisce una stringa contenente l'intero contenuto del file. Inoltre comprende la gestione degli errori e verifica l'esistenza del file.
Il parametro *sPath* indica il percorso del file da leggere.

```
using System;
 using System.IO;
  public string sReadFile(string sPath) {
   string sContent = "";
    try {
     if (File.Exists(sPath)) {
       // apro file in lettura
       StreamReader srTesto = File.OpenText(sPath);
       sContent = srTesto.ReadToEnd();
       srTesto.Close();
      }
    } catch (Exception exRet)
   {
     sContent = "ERRORE: " + exRet.Message;
   }
   return(sContent);
 }
```

CAMBIARE IL COLORE DELLO SFONDO DEI COMPONENTI PRESENTI IN UN FORM

Effettuare questo tipo di operazioni in VC++ è abbastanza complesso, mentre C# dispone di alcune classi che facilitano enormemente il lavoro da svolgere. In effetti basta utilizzare la classe *ColorDialog* e i metodi che questa mette a disposizione.

```
protected void button2_Click (object sender, System.EventArgs e)
{
   ColorDialog colorDlg = new ColorDialog();
   colorDlg.ShowDialog();
   textBox1.BackColor = colorDlg.Color;
   listBox1.BackColor = colorDlg.Color;
   btn.BackColor = colorDlg.Color;
}
protected void button1_Click (object sender, System.EventArgs e)
{
   ColorDialog colorDlg = new ColorDialog();
   colorDlg.ShowDialog();
   textBox1.ForeColor = colorDlg.Color;
   listBox1.ForeColor = colorDlg.Color;
   btn.ForeColor = colorDlg.Color;
}
```

SETTARE LA POSIZIONE DEL CURSORE DEL MOUSE

Questo piccolo programma è in grado di cambiare la posizione del cursore del mouse e di posizionarlo in un punto qualsiasi dello schermo.
Impostazione delle coordinate:

```
        cursor_coord.X=(40-(strlen(buffer)/2));
        cursor_coord.Y=12.5;
```

Nel codice seguente le coordinate sono impostate a (40,12.5).

```
#include <windows.h>
void main(void) {
       HANDLE console_handle;
        COORD cursor_coord;
        char *buffer = "Cursor Position: (40,12.5)";
        DWORD actual=0;
        cursor_coord.X=(40-(strlen(buffer)/2));
        cursor_coord.Y=12.5;
        console_handle= GetStdHandle( STD_OUTPUT_HANDLE);
        if (SetConsoleCursorPosition(console_handle,cursor_coord))
WriteConsole(console_handle,buffer,strlen(buffer),&actual,NULL);
getche(); /* :) */
}
```

SALVARE LA CLIPBOARD IN UN FILE

L'esempio seguente è valido solo per dati in formato testo, ma può essere modificato facilmente intervenendo su *"CF_TEXT"*, cambiando *TEXT* con il tipo di dato presente nella clipboard. Ad esempio nel caso di dati in formato testo basta effettuare una chiamata alla funzione *SaveClipboardTextDataToFile()* seguita dal nome del file: *SaveClipboardTextDataToFile('c:\temp.txt');*.

```
function SaveClipboardTextDataToFile(
  sFileTo : string ) : boolean;
var
  ps1,
  ps2    : PChar;
  dwLen : DWord;
  tf     : TextFile;
  hData : THandle;
begin
  Result := False;
  with Clipboard do
  begin
    try
```

```
      Open;
      if( HasFormat( CF_TEXT ) ) then
      begin
        hData :=
          GetClipboardData( CF_TEXT );

        ps1 := GlobalLock( hData );
        dwLen := GlobalSize( hData );
        ps2 := StrAlloc( 1 + dwLen );
        StrLCopy( ps2, ps1, dwLen );
        GlobalUnlock( hData );
        AssignFile( tf, sFileTo );
        ReWrite( tf );
        Write( tf, ps2 );
        CloseFile( tf );
        StrDispose( ps2 );
        Result := True;
      end;
    finally
      Close;
    end;
  end;
end;
```

DATA E ORA DI CREAZIONE E MODIFICA DI UN FILE

La funzione *FileAge()* restituisce il *date/time stamp* (data e ora di creazione o di modifica) di un file. Il valore ottenuto mediante questa funzione è un intero, prima di essere utilizzato deve essere convertito in un valore *TdateTime (float)*. Il codice seguente mostra come effettuare questa operazione.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  File_Name: string;
  DateTimeStamp: integer;
  Date_Time: TDateTime;
begin
  File_Name := 'c:\Documenti\test.doc';
  DateTimeStamp := FileAge(File_Name);
  // FileAge returns -1 if file not found
  if DateTimeStamp < 0 then
    ShowMessage('File non trovato')
  else begin
    // Converte nel formato TDateTime
    Date_Time := FileDateToDateTime(DateTimeStamp);
    Label1.Caption := DateToStr(Date_Time);
    Label2.Caption := TimeToStr(Date_Time);
  end;
end;
```

ESECUZIONE AUTOMATICA DI PROGRAMMI ALL'AVVIO DI WINDOWS

Questa funzione consente alle applicazioni di essere eseguite all'avvio di Windows. Ad esempio è possibile aggiungere questo codice affinché l'applicazione venga eseguita al successivo avvio di Windows subito dopo l'installazione.

```
procedure RunOnStartup(
  sProgTitle,
  sCmdLine    : string;
  bRunOnce    : boolean
 );
 var
  sKey : string;
  reg  : TRegIniFile;
 begin
  if( bRunOnce )then
    sKey := 'Once'
  else
    sKey := '';

  reg := TregIniFile.Create( '' );
  reg.RootKey:= HKEY_LOCAL_MACHINE;
  reg.WriteString(
    'SoftwareMicrosoft'
    + 'WindowsCurrentVersionRun'
    + sKey + #0,
    sProgTitle,
    sCmdLine);
  reg.Free;
 end;
```

Gli strumenti per costruire Exploit

Metasploit: fai i tuoi Exploit

In questo numero presenteremo una suite valida come ambiente di sviluppo, di test e di esecuzione per qualsiasi tipo di exploit. Parliamo del Metasploit Framework, il sofisticato tool pubblico

Il mondo della sicurezza corre ormai ad una velocità diventata insostenibile anche per grandi compagnie come Microsoft, in perenne affanno nel correre ai ripari ogni qualvolta viene scoperto un nuovo bug nei sistemi operativi che essa produce. Ciò che gli analisti hanno infatti notato è che il tempo medio che intercorre tra il rilascio di un advisory di sicurezza e la disponibilità di un exploit di pubblico dominio va man mano restringendosi, fino ad arrivare, in alcuni casi limite, anche ad una differita di pochi giorni (a volte anche solo 48 ore), come si è visto per alcuni degli ultimi exploit messi in circolazione. Ma se il tempo di creazione degli exploit si è ridotto di gran lunga, un motivo dovrà pur esserci e in questo articolo cercheremo di capire qual è, approfondendo il ciclo produttivo di un exploit dal suo concepimento fino ad arrivare a presentare Metasploit Framework, la suite pensata per i programmatori che devono creare e testare i propri exploit in modo pratico e veloce.

14/Apr
Pubblicazione bollettino Microsoft MS04-011
www.microsoft.com/technet/security/bulletin/ms04-011.mspx

15/Apr
Pubblicazione Exploit per Utility Manager
www.k-otik.com/exploits/04152004.UtilManExploit.c. php

21/Apr
Pubblicazione Exploit per IIS/SSL
www.k-otik.com/exploits/04212004.THCIISSLame.c. php

25/Apr
Pubblicazione Exploit per LSASS
www.k-otik.com/exploits/04252004.ms04011lsass.c.php

01/Mag
Comparsa del worm Sasser, in grado di sfruttare la vulnerabilità di LSASS

Fig. 1: Calendario: dal bug all'exploit in meno di 10 giorni

UNA CORSA PERSA IN PARTENZA...

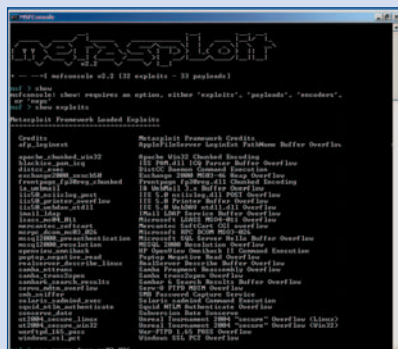
Per testimoniare il continuo rincorrersi tra buoni e cattivi, tra nuovi bug e vecchie patch, basta andare indietro nel tempo e ricordare la storia del bollettino di sicurezza MS04-011, rilasciato in Aprile. Il bollettino nasce per tappare ben quattordici falle di Windows, tra cui vanno ricordate senz'altro quella relativa al servizio di sistema LSASS, quella del server IIS nella gestione di SSL e quella legata all'Utility Manager di Windows (vulnerabilità quasi tutte apparse sulle pagine dei numeri precedenti di ioProgrammo). Facendo i conti a ritroso, si può tracciare il seguente calendario di apparizione di alcuni exploit legati al bollettino MS04-011.
Come si può notare da questo esempio significativo, il rilascio del primo exploit per il bollettino MS04-11 ha richiesto agli hacker circa 24 ore e solo 10 giorni dopo, per tre delle vulnerabilità più critiche dell'advisory, erano presenti in rete i relativi exploit, messi in a mano a ragazzini ansiosi di bucare tutti i server a portata di ping! Il culmine della situazione in quel periodo lo si raggiunge a Maggio con la comparsa del worm Sasser, che nelle sue prime 24 ore di vita riesce ad infettare centinaia di migliaia di host. In questo caso Microsoft sostiene di essere tempestiva nel rilasciare i suoi aggiornamenti, ma dieci giorni – il tempo intercorso dal bollettino alla comparsa del primo exploit critico per LSASS – sono un tempo ridicolo per un sistemista messo alle strette, che prima di applicare una patch che andrà ad alterare il funzionamento dei propri sistemi in produzione, necessita di tempo aggiuntivo per effettuare le opportune verifiche di compatibilità.
Se poi si aggiunge il fatto che spesso alcune patch non

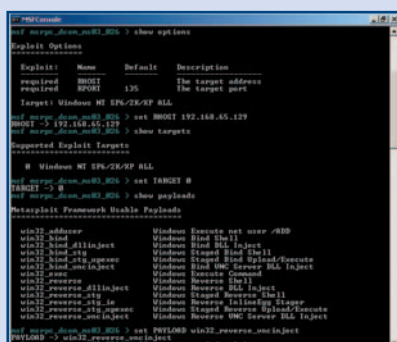
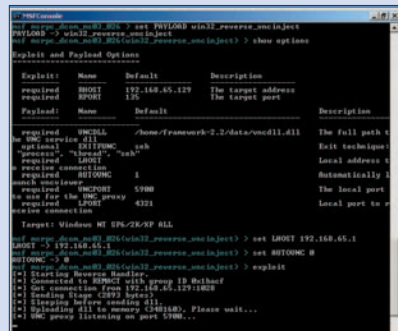
AGGIORNAMENTI DEGLI EXPLOIT MSF

Il framework di Metasploit può essere aggiornato via web con nuovi moduli e nuovi exploit periodicamente pubblicati sul sito www.metasploit.org o in alternativa si possono installare attraverso il collegamento del menù "MSFUpdate", che provvede in modo automatico a scaricare i nuovi componenti da metasploit.org per installarli. Una sorta di WindowsUpdate....degli hacker!

sono testate a dovere per la fretta del rilascio e che possono rivelarsi più dannose che utili, viene da chiedersi se sia davvero il caso di aggiornare il sistema. Sempre prendendo spunto dalla realtà, il Service Pack 2 di Windows XP che in queste ore imperversa sui PC di mezzo mondo, stando ai primi rapporti, ha causato più problemi di un virus, a causa delle sue restrizioni di sicurezza spropositate e delle sue enormi incompatibilità con numerosi programmi come antivirus, firewall, client di rete, che dopo l'installazione del "pacco" smettono di funzionare all'istante!

SIMULAZIONE DI UN EXPLOIT ATTRAVERSO LA CONSOLE MSF

Nell'esempio, documentato passo dopo passo, viene mostrato come è facile lanciare un exploit del servizio RPC/DCOM (MS04-043) contro un server Windows 2000, attraverso la console del Metasploit Framework.

1 **Una volta lanciata, la console MSF appare come in figura. Per avere l'elenco degli exploit caricati bisogna usare il comando *"show exploits"* e scegliere il modulo desiderato tramite il comando *"use nome_modulo"*. Per ottenere la lista delle opzioni configurabili nel modulo scelto basta scrivere *"show options"*.**

2 **Nell'exploit per RPC la porta remota (RPORT) è già settata su 135, quindi non rimane che impostare l'indirizzo IP dell'host attaccato usando il comando *"set RHOST indirizzo_ip"*. Il comando *set* imposta qualsiasi altra opzione. Similmente con *"show targets"* otteniamo i sistemi vulnerabili a questo attacco e usando *"set TARGET 0"* impostiamo il nostro bersaglio.**

3 **Dopo aver settato il target bisogna scegliere il payload fra quelli elencati da *"show payloads"*, che rappresentano i diversi moduli del framework. Nell'esempio vogliamo provare una tecnica avanzata di DLL injection, capace di attivare un server VNC sull'host vittima, ritornando il controllo totale sul sistema remoto. Il payload si imposta sempre tramite *"set PAYLOAD nome_payload"*.**

4 **Al termine della configurazione del payload (sempre tramite il comando *"show options"* e *"set"*) si lancia l'attacco digitando *"exploit"*. Se tutto va bene, al termine dell'attacco potremo collegarci usando un VNC client all'host remoto sulla porta 5900, che ci aprirà il controllo totale del sistema remoto (in questo esperimento la vittima è un server Windows 2000).**

THINK LIKE AN ATTACKER!

Tornando all'analisi del fenomeno di contrazione dell'intervallo "bollettino di sicurezza -> exploit", c'è da dire che esso porta con sé due conseguenze spiacevoli: da un lato lascia agli amministratori e ai sistemisti poco tempo per difendersi ed applicare le patch in grado di arrestare l'avanzata di hacker e worm, dall'altro i produttori devono mettere in piedi un'infrastruttura di rilascio aggiornamenti talmente efficiente e veloce, da restare al pari col nemico! Se poi si verifica che il worm o l'exploit di turno viene rilasciato – per pura combinazione o volutamente – durante un week-end (il venerdì sera sembra essere il giorno preferito dagli hacker…) allora si corre il rischio di restare scoperti per più di 48 ore e di trovare al riento la propria rete infestata. Quali le cause di questa situazione? Il restringersi dell'intervallo è dovuto a diversi fattori, tra cui senz'altro l'incremento della competenza di chi produce gli exploit e il ri-utilizzo del codice già prodotto da altri hacker.

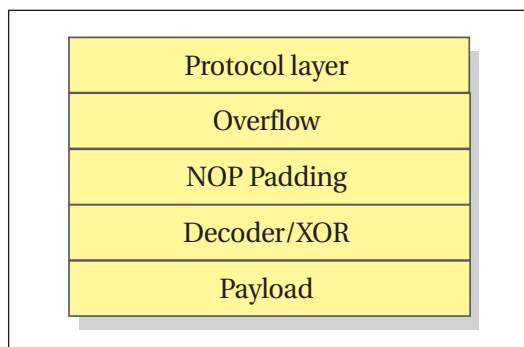



Fig. 2: ***Struttura di un generico exploit***

È infatti stato appurato che le parti di codice comune tra exploit diversi sono più di uno e che, seguendo la filosofia di "riciclo del codice" molto comune nel mondo della programmazione, anche gli hacker si siano adeguati producendo codice modulare, componibile e facilmente riutilizzabile. D'altronde gli exploit sono e restano pur sempre dei programmi… e come tali sono soggetti alle leggi del copia&incolla!
Proprio seguendo questo approccio di programmazione modulare è stato concepito il Metasploit Framework, riflettendo sul fatto che un exploit può essere schematizzato e scomposto in alcuni elementi atomici riutilizzabili.

Se prendiamo in esame un qualsiasi exploit, vedremo che infatti può essere suddiviso in diverse sezioni, alcune delle quali spesso sono comuni.
Il primo layer (relativo al protocollo di trasmissione) è tipico degli exploit remoti, dove prima di iniettare il codice malevolo è indispensabile stabilire una comunicazione con l'host attaccato. Questo livello generalmente contiene un pacchetto formattato secondo il protocollo di comunicazione richiesto (RPC, netbios, http, telnet, ssh, ftp, ecc.) con un header ben definito e può variare da exploit a exploit. In genere, integrato in questo pacchetto (o immediatamente dopo), viene collocato l'overflow di bytes – ad esempio una sequenza di 1000, 2000 o più caratteri uguali, il cui scopo è causare il sovraccarico dell'area dello stack nel caso di exploit di tipo "buffer overflow" (BOF). Per i dettagli su questo tipo di overflow si rimanda all'articolo in merito pubblicato su ioProgrammo numero 72 (Sett. 2003).
L'overflow può essere seguito da un tappeto di NOP, ovvero una sequenza molto lunga di istruzioni ininfluenti (NOP è l'istruzione assembly di no-operation) che servono a compensare i problemi di indeterminazione sull'indirizzo di arrivo dell'exploit. L'hacker infatti non sa con precisione dove atterrerà il codice del suo exploit così, spesso, inserisce un tappeto di istruzioni NOP che gli danno un certo margine di errore.
Prima del payload, ovvero il cuore dell'exploit, possiamo trovare (ma non è indispensabile) un decoder che serve a decriptare il codice del payload. Questo meccanismo, realizzato spesso tramite semplici istruzioni di XOR, serve a mascherare gli exploit sotto gli occhi dei sistemi anti-intrusione (IDS) ed evita il problema dei bytes nulli (0x00), che spesso non possono essere trasmessi in quanto filtrati.

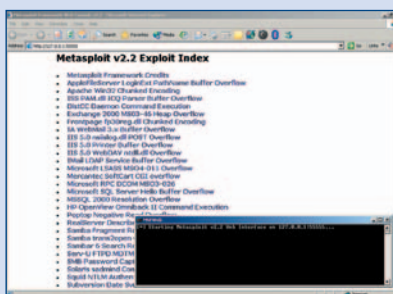
A CIASCUNO IL SUO...PAYLOAD!

Payload è spesso usato come sinonimo di "shellcode", ma non è sempre così. Con la parola payload s'intende il codice finale che l'exploit riuscirà ad eseguire sul sistema remoto per ottenere un accesso non autorizzato. Lo shellcode è quel particolare payload che ha il compito di aprire una shell sul sistema remoto ("/bin/sh" su sistemi Unix, "cmd.exe" su Windows per intenderci) ma esistono una miriade di altri payload diversi, per cui è meglio fare un po' d'ordine organizzando le idee sui payload secondo queste principali categorie:

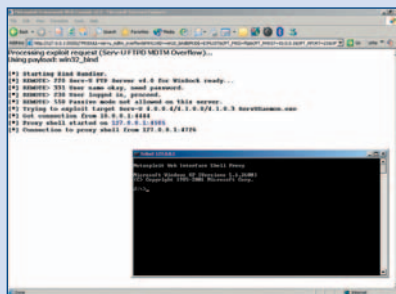
- **bindshell** - l'exploit apre una shell di sistema e la pone in ascolto su una certa porta dirigendo l'I/O su una socket; dopo aver lanciato l'exploit, all'hacker non resta che collegarsi via telnet alla porta di ascolto aperta;
- **reverseshell** - l'hacker prima di eseguire l'exploit, apre una porta in ascolto sul proprio host tramite l'utility NETCAT; all'esecuzione dell'exploit, l'host attaccato aprirà una shell e sarà forzato a connettersi alla porta d'ascolto ospitata sull'host dell'hacker; questo meccanismo è molto utile per quegli host che sono protetti da un firewall, dove non funzionano i "bindshell";

SIMULAZIONE DI UN EXPLOIT ATTRAVERSO L'INTERFACCIA WEB

Il framework non ha solo la modalità console, ma ha anche di una comoda interfaccia web che permette di lanciare tutti gli exploit in modo comodo, senza dover ricordare a memoria i comandi della console. Nell'esempio lanceremo un attacco contro un server FTP Serv-U 4.x che mira ad aprire una bindshell.

1 **L'interfaccia web della console si attiva lanciando il server web di MSF tramite il collegamento nel menù chiamato *"MSFWeb"*. Il server viene attivato sulla porta 55555 e può essere raggiunto tramite un qualsiasi browser all'url locale http://127.0.0.1:55555, dove viene riportato l'elenco degli exploit caricati nel framework.**

2 **In questo caso scegliere un payload adatto fra quelli riportati è un gioco da ragazzi; i moduli di payload sono simili a quelli visti nell'altro esempio di exploit e possono essere scelti con un semplice click. Per la nostra simulazione, useremo un payload di tipo *"bindshell"*.**

3 **Tutte le configurazioni fatte a mano nella console, come gli indirizzi IP, le porte e le altre opzioni, nell'interfaccia web sono tutte disponibili nella pagina e possono essere compilate con calma e in modo semplice. Per testare l'exploit, al termine della configurazione, basta fare clic su *"Launch Exploit"*.**

4 **Una volta eseguito l'exploit, l'output generato viene rediretto sull'interfaccia web dove è possibile seguire passo-passo l'attacco. Nell'ultima riga vediamo che l'exploit ha avuto successo e che una bindshell è disponibile all'indirizzo "127.0.0.1:4505" dove possiamo eseguire telnet per collegarci e prendere controllo del sistema.**

NOTA

UN SERVICE PACK INDIGESTO

Le incompatibilità prodotte del Service Pack 2 di Windows XP con i normali programmi che siamo soliti usare (antivirus, firewall, client ftp, programmi di rete, remote desktop) sono numerose e possono essere spulciate al seguente link (in francese), fornito dal portale di IT Security K-Otik. www.k-otik.com/news/08152004.XPSP2Incompatible.php

La battaglia infinita tra buoni e cattivi...
Il sito www.wired.com/wired/archive/12.09/view.html?pg=3 **ha riportato qualche tempo fa una simpatica intervista a Stephen Toulouse, il Security Program Manager di Microsoft, interrogato riguardo l'eterna lotta tra bene e male. Lo scambio di battute è stato il seguente e quasi testimonia lo sconforto di S. Toulouse nel combattere gli hackers e nel difendersi dai bug:**

D.: "Non le sembra di combattere una battaglia già persa?"

R.: "Non si tratta di un interruttore che può essere spento. Il software è creato dagli uomini e conterrà sempre errori..."

SUL WEB

LINK

Tutorial sugli shellcode per beginners
www.vividmachines.com/shellcode/shellcode.html

Crash Course User Guide per il Metasploit Framework
http://metasploit.com/projects/Framework/docs/CrashCourse-2.0.pdf

Shellcode Advanced
www.securitydate.it/SD2004/slides/shellcode/pdf /shellcode-advanced.pdf

LSD (Last Stage of Delirium) – Win32 Assembly Components
www.lsd-pl.net/ documents/winasm-1.0.1.pdf

- **adduser** - l'exploit esegue una chiamata che aggiunge un utente a scelta al sistema, coi privilegi di Administrator;

- **exec** - l'exploit viene progammato per eseguire un qualsiasi comando di sistema, potenzialmente anche in grado di distruggere o compromettere dati.

Il payload in genere è la parte più riciclata e scambiata fra gli hacker, proprio perché può essere programmata in modo indipendente dal resto dell'exploit. E' capitato infatti in passato che alcuni exploit innocui e senza alcuno scopo malevolo, pubblicati dai ricercati solo per dimostrare la presenza di una vulnerabilità (chiamati in gergo PoC, proof-of-concept), siano stati ripresi e modificati con l'aggiunta di payload già esistenti, diventando armi micidiali in mano a ragazzini neanche troppo esperti. Ad un tratto infatti realizzare un exploit diventa sempre più simile a comporre un puzzle i cui pezzi sono reperibili oramai ovunque su Internet. I problemi tipici che si possono incontrare nella creazione di un exploit sono i seguenti:

- **Function Address variabili** – un exploit pensato per funzionare su Windows XP Home non funziona su XP Professional e allo stesso modo non funziona su Windows 2000. Spesso anche tra sistemi operativi identici, ma in lingue diverse, si riscontra questa differenza di funzionamento degli exploit. Ciò dipende dal fatto che le funzioni di sistema usate da un exploit vengono esportate direttamente dalle DLL di Windows; tali librerie (kernel32, user32, ntdll, ecc.) cambiano, seppure di poche centinaia di bytes, tra le diverse release di Windows, rendendo difficile la creazione di exploit davvero universali. Per ovviare a questo problema alcuni hacker (il team conosciuto come LSD) hanno scritto un codice di particolare codice assembly in grado di andare a scovare, nei meandri della memoria, le funzioni esportate dalle DLL, rendendo qualsiasi exploit capace di operare su qualsiasi ambiente.

- **Dimensione del payload** – il payload non può essere grande a piacere, perché spesso i pacchetti inviati non possono superare alcune dimensioni o perché lo stack dove saranno iniettati non è sufficientemente grande per contenerli. Gli hacker in questi casi si trovano a lottare per risparmiare manciate di bytes tagliando istruzioni qua e là nel codice del payload, qualche tempo fa è addirittura apparsa qualche gara nella scrittura dello shellcode più corto (che si assesta intorno ai 25-30 bytes al momento).

Con Metasploit Framework è possibile risolvere tutti questi problemi e creare exploit universali.

METASPLOIT FRAMEWORK ALL'OPERA

Terminata la disquisizione teorica, è il caso di passare alla presentazione del Metasploit Framework (MSF) più da vicino, che può essere considerato il risultato di tutti i nostri discorsi sul recente proliferare di exploit e sul riutilizzo dei codici all'interno di questi.
MSF è una suite completa per le creazione e il test di exploit totalmente scritta in linguaggio Perl, con qualche contaminazione di Assembly e C++ per le parti più "hard". Gli autori sono il team di hackers conosciuto col nome di Metasploit (*www.metasploit.org*). Lo scopo principale di questa incredibile suite è uno soltanto: mettere a disposizione di chiunque un ambiente di sviluppo ed esecuzione degli exploit totalmente modulare, programmabile ed estensibile in qualsiasi momento. La suite è infatti formata da una moltitudine di componenti indipendenti tra di loro, ogni componente è ri-utilizzabile ed inoltre l'intero framework è totalmente open-source; ciò significa ad esempio che se dovessimo imbatterci nell'ennesimo problema di buffer overflow di qualche sistema, sarebbe davvero banale creare un exploit funzionante in poco tempo, sfruttando i moduli già pronti del framework.
A fronte di questa considerazione possiamo intuire facilmente il perché il tempo di comparsa dei nuovi exploit si è ridotto notevolmente; con MSF anche chi non ha molta dimestichezza con shellcode e buffer overflow, può improvvisarsi hacker! È come disporre quindi di una sorta di SDK (completo di librerie di payload già pronti all'uso) per la creazione dei propri exploit.

Elia Florio

Dal mondo reale al digitale: monitorare fenomeni fisici

Un voltmetro di precisione in Delphi

Controllare il mondo esterno con il vostro PC: una guida per apprendere come sia possibile gestire parametri di qualunque natura per mezzo di un voltmetro di precisione controllato da PC

Il mondo che ci circonda è regolato da meccanismi di varia natura, soprattutto fisici, chimici e biologici che influenzano la nostra vita di tutti i giorni. La naturalezza di questi meccanismi ed il fatto che ne siamo immersi in modo continuativo, fanno sì che ormai non ne notiamo quasi più i cambiamenti, salvo che queste non superino il loro normale campo di variazione. Il caso estremo si verifica con l'esempio che spesso viene citato nei corsi di sociologia: la "rana billata". Se poniamo una rana in una pentola di acqua fredda e procediamo ad innalzarne gradualmente e lentamente la temperatura, la nostra povera amica si ritroverà bollita senza neppure rendersene conto. Che cosa questo abbia a che fare con la programmazione e l'elettronica è presto detto: si dimostra che il controllo di ogni parametro del mondo reale necessita di una adeguata rilevazione e gestione in modo analogico, oppure di un campionamento e controllo digitale per mezzo di una scala ed una risoluzione appropriati. Svilupperemo una applicazione completa di un voltmetro di precisione per PC, accennando a come sia possibile acquisire un qualunque parametro fisico. Il voltmetro presentato in queste pagine trasformerà il nostro computer in un sistema completo di controllo. Sarò ovviamente a disposizione dei docenti e degli studenti che vogliano approfondire l'argomento sul forum di ioProgrammo, oppure attraverso l'indirizzo di posta elettronica *luca.spuntoni@ioprogrammo.it.*

UN PROBLEMA DI TRASPOSIZIONE DEI PARAMETRI

Supponiamo di volere controllare l'andamento della temperatura di un determinato processo industriale: innanzitutto occorre definire il campo di misura della temperatura. Ipotiziamo ad esempio che l'escursione della temperatura del processo vari nel campo 0-100 gradi centigradi. Il passo successivo consiste nello stabilire il livello di risoluzione che occorre al fine della misurazione della temperatura. Supponiamo di desiderare una risoluzione di 0,5 gradi centigradi nel campo di misura in questione, questo comporta che il numero di valori possibili è di 100/0,5=200, comportando il fatto che sono sufficienti otto bit per rendere possibile la misura con la risoluzione prefissata, nel campo di variazione del parametro in questione. Con otto bit sono possibili $2^8=256$ livelli di misura, quindi a questo punto occorre stabilire se aumentare la risoluzione della misura, oppure espanderne il campo per sfruttare tutti i 256 possibili valori. Supponiamo ancora di scegliere di espandere il campo di misura, potremmo ad esempio scegliere di misurare il campo compreso tra -10 e + 127 gradi centigradi con una risoluzione di 0,5 gradi. Detto questo, poiché il nostro convertitore analogico-digitale misura una tensione elettrica, nel camp 0-5 Volts, sarà necessario un semplice amplificatore, collegato ad un sensore di temperatura tarato in modo tale da fornire 0 V al limite inferiore del campo di misura (-10 gradi nel nostro esempio) e 5 Volts al limite superiore, corrispondenti a +127 gradi dell'applicazione in questione.

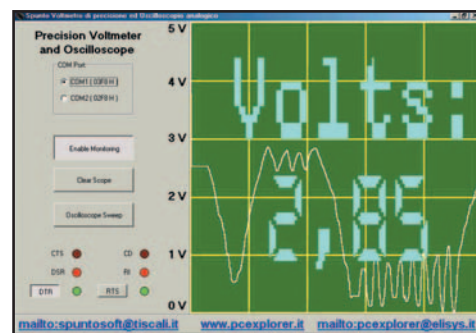
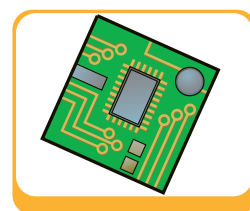
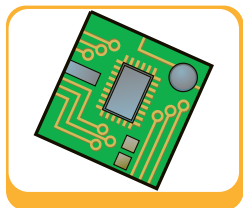



Fig. 1: Il software del Voltmetro per PC è presente nel CD allegato alla rivista completo di codice sorgente

SCHEMA ELETTRICO

La moltiplicazione delle linee di ingresso

Il nostro voltmetro di precisione è basato sul convertitore analogico-digitale ADC804, descritto nel dettaglio nel numero precedente della rivista. La filosofia di implementazione del sistema consiste nello sfruttare le quattro linee di ingresso asincrono della porta seriale (CTS,DSR,CD e RI) al fine di poter

COMPONENTI ELETTRONICI NECESSARI

N 1 ADC804
N1 4019
N1 4011
N2 Transistor BC 107
N1 Res. 5600 Ohm_Watt
N7 Res. 10 k Ohm_Watt
N1 Potenziometro 10K Ohm Lineare
N1 Condensatore 1uF 16V
N1 Condensatore 150 pF
N1 Connettore 9 poli femmina DB9
N1 Basetta millefor
www.pcexplorer.it

leggere le otto linee di uscita D0-D7 del convertitore analogico-digitale. Ma come si possono leggere otto linee di uscita per mezzo di sole quattro linee di ingresso? Utilizzando il circuito integrato 4019, dotato di multiplexer, insieme al componente 4011, contenete quattro porte logiche NAND è possibile compiere questo piccolo miracolo. Per compiere la lettura dei due nibbles (D0-D3 e D4-D7) è sufficiente variare lo stato logico della linea DTR della porta seriale. In questo modo quando DTR è allo stato logico HIGH, sulle quattro linee di ingresso della porta seriale saranno disponibili D3-D7, mentre quando si trova allo stato logico LOW verranno letti D0-D3. I gruppi R1-R2-TR2 e R3-R4-TR2 sono necessari per operare la conversione tra il livello logico RS232, che può variare anche fino a +-25 Volts, e lo standard TTL utilizzato sulla nostra applicazione (0-5 Volts). In particolare TR1 opera la selezione dei due gruppi di quattro bit del convertitore analogico-digitale, mentre TR2 si occupa di attivare la conversione A/D. Osservando lo schema elettrico, sul lato sinistro notiamo le linee di connessione della porta seriale, prelevate per comodità dall'apparecchiatura PCExplorer light, dotata di un'adeguata alimentazione interna, senza la necessità di alcuna saldatura. Il circuito può essere ugualmente realizzato prelevando le linee in questione per mezzo di un comune cavo seriale. Il circuito è in grado di misurare segnali di tensione compresa tra 0 V e +5 V, ogni tensione al di fuori di questi valori può danneggiare il convertitore, o peggio i circuiti interni del PC: si raccomanda di rispettare strettamente questi limiti. La conversione inizia quando viene inviato un impulso *LOW* alla linea /WR mentre l'ingresso /CS (Chip Select) è a livello logico *LOW*. La lettura degli otto bit contenuti nella *LATCH* può essere ottenuta inviando un impulso sulla linea /RD mentre l'ingresso /CS è a livello logico LOW. Per maggiori informazioni sul componente, per ragioni di spazio, si consiglia la lettura del documento citato in bibliografia, disponibile sul WEB presso il sito della Philips Semiconductors.

***Fig. 2:** Le connessioni dell'integrato ADC804*

PRECAUZIONI

Prima di collegare il circuito al nostro PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato come previsto. Il circuito è in grado di misurare segnali di tensione compresa tra 0 V e +5 V, ogni tensione al di fuori di questi valori può danneggiare il covertitore, o peggio i circuiti interni del PC.

'MULTIPLEXER' 4019

Il circuito integrato 4019 comprende al proprio interno quattro multiplexer, dotati di due ingressi comuni di selezione, chiamati SA ed SB. A questo scopo ciascun circuito di multiplexing contiene due ingressi An e Bn, nonché una sola uscita che viene chiamata On. L'utilizzo generale di questo circuito integrato è quello di selezionare, uno per volta, due gruppi di quattro bit: per fare ciò in uscita troviamo i valori dei bit A quando SA si trova a livello logico alto, mentre troviamo i bit B quando SB è a livello logico alto. Se entrambi i bit di selezione sono a livello logico alto, in uscita troviamo il risultato della operazione di OR logico tra i bit A ed i bit B. In ultima analisi se entrambi i segnali SA ed SB sono a livello logico LOW, in uscita troviamo un livello LOW, indipendentemente dallo stato degli ingressi A e B.

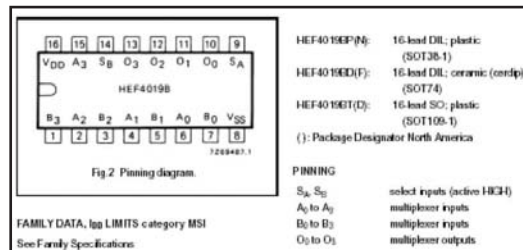
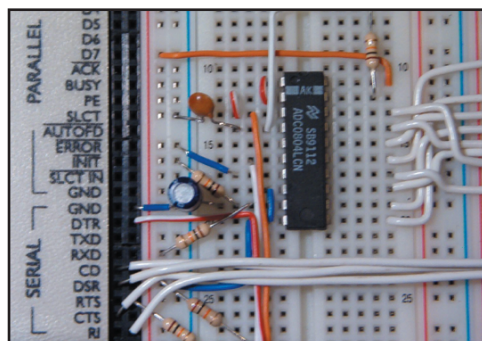



***Fig. 3:** In figura si riporta la piedinatura del circuito integrato HEF4019, reperibile in forma completa su Internet all'indirizzo: www.components.philips .com/ (cortesia Philips Semiconductors)*

PORTE LOGICHE NAND L'INTEGRATO 4011

Esiste una famiglia di circuiti integrati che permette di eseguire operazioni logiche autonomamente, senza l'ausilio del microprocessore. Volendo fare un esempio pratico, supponiamo di avere due pulsanti, che chiamiamo A e B, ed una lampadina che identifichiamo con Y: inoltre per nostra convenzione stabiliamo che la pressione del pulsante e l'accensione della lampadina corrisponda ad un livello logico "1", mentre il contrario ad un livello logico "0". Nella *Tabella 1* che segue, detta "tabella della verità" si ha un riassunto dei risultati logici delle operazioni logiche *NOT, OR* ed *AND,* in aggiunta a queste, molto utilizzate in elettronica, troviamo anche *NOR (NOT OR)* e *NAND (NOT AND),* che si riferiscono semplicemente alla negazione logica delle operazioni *OR* ed *AND*.

***Fig. 4:** Nell'immagine di figura si riporta lo schema logico e la piedinatura del circuito integrato HEF4011, reperibili in forma completa su Internet all'indirizzo: www.components.philips.com (cortesia Philips Semiconductors)*

Di seguito si riportano le connessioni dell'integrato 4011, contenente al proprio interno quattro porte

logiche *NAND* a due ingressi *(cortesia Philips Semiconductors)*, gli ingressi sono chiamati I1-I8, mentre le uscite O1-O4; è da notare che l'operazione logica effettuata da una porta non influisce assolutamente con quelle delle altre. Affermavamo che le porte *NAND* e *NOR* sono molto più diffuse delle porte *AND* ed *OR* e ci si potrebbe chiedere perché, dal momento che le seconde sono, in effetti le operazioni logiche fondamentali. Il motivo è essenzialmente pratico: osservando il simbolo della porta logica, si nota un pallino (il simbolo della negazione logica) posto in prossimità del terminale di uscita: significa che l'operazione logica che avviene all'interno della porta è *Y=NOT(A AND B)*, quindi se colleghiamo elettricamente A e B insieme (ponendo quindi *A=B*) la relazione precedente diventa *Y=NOT(A AND A)*, ovvero *Y=NOT A:* abbiamo costruito una porta logica *NOT*, collegando gli ingressi di una porta *NAND*. La maggiore diffusione di porte *NAND* si spiega con la loro maggiore flessibilità, dal momento che permettono di ottenere, con un solo microchip dotato di quattro porte, tutte le configurazioni logiche.

REALIZZAZIONE DEL VOLTMETRO

Il circuito elettronico può essere realizzato senza saldature per mezzo dell'apparecchiatura PC Explorer light, ma anche con i soli componenti elettronici elencati a lato di queste pagine, per mezzo del loro montaggio su una comune basetta millefori, utilizzando un comune saldatore a stagno. L'assemblaggio dei componenti viene facilitato osservando le immagini riportate in queste pagine, disponibili anche nel file allegato alla rivista. Si consiglia di inserire prima i componenti a profilo più basso, iniziando con il circuito integrato e con le resistenze, per poi collegare i condensatori ed il transistor. Le connessioni possono essere realizzate per mezzo di spezzoni di filo elettrico rigido.

IL SOFTWARE DEL VOLTMETRO

Per motivi di brevità analizzeremo, come di consueto, soltanto le parti salienti del software di controllo, per ogni eventuale chiarimento, delucidazione o consiglio, l'autore è lieto di rispondere a qualunque domanda. Il codice sorgente, scritto in Delphi viene messo a completa disposizione del lettore ed è disponibile nel CD con il nome: *"SpuntoVoltmetro-Oscilloscopio.zip"*. La procedura che segue rappresenta l'event handler dell'oggetto *Timer1*, che provvede ad innescare l'esecuzione del codice ad intervalli prestabiliti dall'utente, per default questo intervallo di tempo è di un secondo, ma può essere agevolmente variato modificando la proprietà *'Interval'* dell'oggetto in questione. La gestione dell'hardware avviene azionando le due linee di uscita della porta seriale *RTS* e *DTR*, rispettivamente per generare il segnale di clock per il convertitore analogico-digitale e per selezionare il gruppo di quattro bit (D0-D3 e D4-D7) che si desidera leggere per ricostruire il byte

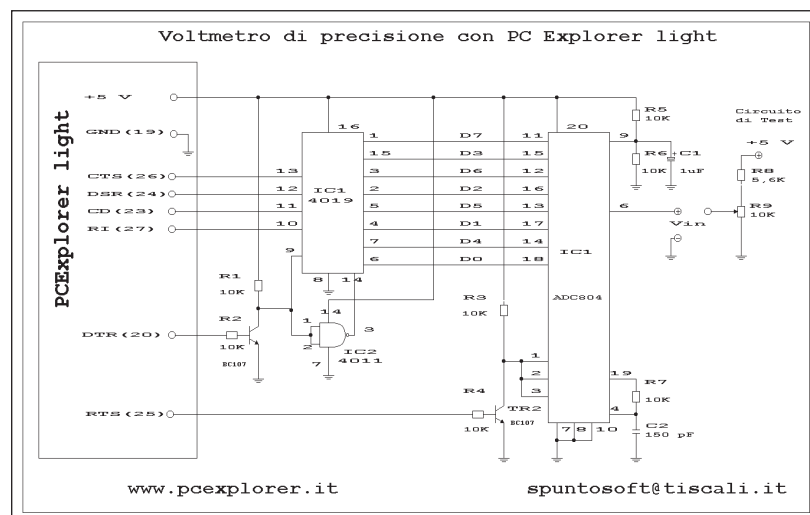
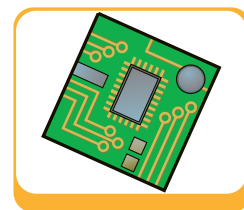



Fig. 5: ***Lo schema elettrico è disponibile nel file allegato alla rivista (SpuntoVoltmetroOscilloscopio.zip)***

corrispondente al valore binario generato dal convertitore. Le quattro linee di ingresso della porta (CTS, DSR, CD e RI) invece, provvedono a leggere i valori di ogni singolo bit provenienti dal convertitore attraverso i multiplexer contenuti nel circuito integrato 4019.

```
procedure TSpuntoVoltmetroOscilloscopioForm
                         .Timer1Timer(Sender: TObject);
VAR
MCRWord:Word;
begin
ADCValue:=0;
ReadAllPorts;
If RTS then LedRTS.LedOn else LedRTS.LedOff;
If CTS then LedCTS.LedOn else LedCTS.LedOff;
If DSR then LedDSR.LedOn else LedDSR.LedOff;
If CD  then LedCD.LedOn else  LedCD.LedOff;
If DTR then LedDTR.LedOn else LedDTR.LedOff;
If RI  then LedRI.LedOn else  LedRI.LedOff;
```

Fino a questo punto della procedura abbiamo letto le linee della porta seriale ed abbiamo impostato i corrispondenti oggetti sulla form. Successivamente

CODICE ALLEGATO ALLA RIVISTA

Il codice sorgente e tutti i componenti necessari sono reperibili sul CD allegato alla rivista all'interno del file: *SpuntoVoltmetro-Oscillo scopio.zip.*

A	B	Y= NOT A	Y= NOT B	Y=A OR B	Y=A AND B	Y=A NOR B	Y=A NAND B
0	0	1	1	0	0	1	1
0	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1
1	1	0	0	1	1	0	0

TABELLA 1: Le possibili combinazioni binarie

provvediamo a generare il clock per il convertitore attraverso la variazione dello stato logico della linea RTS ed a selezionare il nibble più significativo impostando DTR a livello logico *High (DTR=True)*. Successivamente la codifica delle linee di uscita della porta vengono inviate all'indirizzo fisico della seriale *(WritePort(MCRAddress,MCR Word))*.

```
ReadAllPorts;
MCR[1]:=NOT(PresentRTSStatus); // RTS Clock
                                    per il convertitore
MCR[0]:=True; // DTR=True Nibble più significativo D4-D7
//Codifico MCR in Word e poi lo invio alla porta
EncodePortArray(MCRWord,MCR);
WritePort(MCRAddress,MCRWord);
PresentRTSStatus:=MCR[1];
 If Not(PresentRTSStatus) then begin
  Image1.Canvas.Pen.Color:=clWhite;
  Image1.Canvas.Pen.Width:=2;
```

Successivamente avviene la conversione del nibble più significativo, come analisi del singolo bit di ingresso della porta seriale.

```
Sweep:=Sweep+1;
//CTS
if CTS Then begin
ADCValue:=ADCValue+128;
end;
//DSR
if DSR Then begin
ADCValue:=ADCValue+64;
end;
//CD
if CD Then begin
ADCValue:=ADCValue+32;
end;
//RI
if RI Then begin
ADCValue:=ADCValue+16;
end;
```

A questo punto si opera la selezione del nibble meno significativo (D0-D3), ponendo DTR a livello logico *Low (DTR= False)* e si provvede ad impostare nuovamente le linee di uscita per mezzo della procedura *WritePort*.

```
ReadAllPorts;
MCR[0]:=False; // DTR=False Nibble meno
                               significativo D0-D3
//Codifico MCR in Word e poi lo invio alla porta
EncodePortArray(MCRWord,MCR);
WritePort(MCRAddress,MCRWord);
PresentRTSStatus:=MCR[1];
ReadAllPorts;
```

La decodifica del nibble meno significativo (D0-D3) avviene di seguito.

```
//CTS
if CTS Then begin
ADCValue:=ADCValue+8;
end;
//DSR
if DSR Then begin
ADCValue:=ADCValue+4;
end;
//CD
if CD Then begin
ADCValue:=ADCValue+2;
end;
//RI
if RI Then begin
ADCValue:=ADCValue+1;
end;
```

Una volta terminata la decodifica dell'intero byte proveniente dal convertitore A/D, vengono calcolati i valori di tensione, nonché le coordinate per la successiva rappresentazione sul display e sullo schermo grafico del programma.

```
//ADC Value
ADCValueY:=Round((500-(500/256)*ADCValue));
Volts:=ADCValue*5/256;
VoltsLabel.Caption:=FloatToStrF(Volts,ffGeneral,3,5);
If (OscilloscopeSweep.Down and (Sweep<500)) then
                                              Begin
 Image1.Canvas.MoveTo(Sweep,(CurrentY));
 Image1.Canvas.LineTo((Sweep+1),(ADCValueY));
 CurrentY:=ADCValueY;
End
Else Begin
 OscilloscopeSweep.Down:=False;
 Sweep:=0;
End;
End;
end;
```

Sono stati utilizzati due componenti sviluppati con Delphi (*SpuntoLedComponent* e *SpuntoHyperLabel*) che sono distribuiti nel file allegato al CD della rivista in forma compilata: chiunque desideri ricevere il codice sorgente di questi componenti può richiederli all'indirizzo *spuntosoft @tiscali.it*. L'installazione e l'esecuzione del programma non creano difficoltà, il software viene fornito anche nella versione completamente compilata e collaudata (*SpuntoVoltmetro.exe*).

L'AVVIO IN WINDOWS

L'esecuzione del programma non presenta grosse difficoltà, tutti i comandi sono disponibili in

ACQUISTARE I COMPONENTI ELETTRONICI

Il lettore potrà reperire i pochi componenti elettronici che utilizzeremo per la costruzione dell'oscilloscopio, in qualunque negozio di componenti elettronici, oppure per corrispondenza presso la Elisys s.r.l. www.pcexplorer.it. Si è resa disponibile ad offrire in omaggio il kit di realizzazione del voltmetro / oscilloscopio a chi acquisterà una apparecchiatura PCExplorer light.

L'autore è lieto di rispondere ai quesiti dei lettori sull'interfacciamento dei PC all'indirizzo: luca.spuntoni@ioprogrammo.it

un'unica form: partendo dall'alto notiamo i radio buttons deputati alla selezione della porta seriale alla quale è collegato il circuito elettronico, con i relativi indirizzi di default per COM1 e COM2.
Nell'eventualità in cui i parametri non coincidano con quelli del computer che si sta utilizzando, il lettore non avrà difficoltà di modificarne il valore nelle procedure *RadioButtonCOM1click* e *RadioButtonCOM2click*.
Il pulsante "*Enable Monitoring*" provvede ad abilitare la lettura periodica dei valori di tensione attraverso il convertitore A/D, mentre i pulsanti "*Clear Scope*" e "*Oscilloscope Sweep*" provvedono rispettivamente a pulire lo schermo dell'oscilloscopio ed ad abilitarne la scansione. Di seguito notiamo i led relativi alle quattro linee di ingresso asincrono della porta seriale (CTS, DSR, CD e RI), nonché i due pulsanti ed i due LED delle linee d'uscita della porta (DTR ed RTS). Sul lato destro della finestra notiamo lo schermo dell'oscilloscopio, mentre in sovrimpressione il valore di tensione elettrica misurata dal circuito espressa in Volt. Infine, in basso notiamo alcuni link per mezzo del quale è possibile ricavare molte informazioni utili sull'interfacciamento di circuiti elettronici per mezzo del Personal Computer.
L'utilizzo del programma su sistemi dotati di Win 2000, XP oppure NT, è possibile seguendo le istruzioni riportate di seguito. Al fine di consentire l'esecuzione del software, poiché questo accede direttamente all'hardware della macchina, è necessario installare il driver: "*PortTalk - A Windows NT/2000/XP I/O Port Device Driver Version 2.2*" scaricabile all'indirizzo: *www.beyondlogic.org /porttalk /porttalk.htm* e contenuto nel CD allegato alla rivista. Il file "*Porttalk22.zip*" contiene tutte le istruzioni necessarie all'utilizzo corretto del driver, nonché una notevole mole di informazioni relative all'accesso delle porte hardware del PC: viene fornito inoltre il codice sorgente completo delle applicazioni.
Per quanto riguarda l'utilizzo del programma proposto in questa sede sotto Win 2000/NT/XP è sufficiente estrarre i file contenenti il driver "*Porttalk*" nella directory del programma da utilizzare e "chiamare" l'applicazione *"SpuntoVoltmetro.exe"* per mezzo del comando: *"C:\>Allowio SpuntoVoltmetro.exe/"* che consente l'accesso da parte dell'eseguibile "*SpuntoVoltmetro.exe*" a tutte le porte del PC.

LA RILEVAZIONE DEL POTENZIOMETRO

Per mezzo dell'esempio che proponiamo qui di seguito è possibile operare la lettura di un qualunque sensore a resistenza variabile, come ad esempio fotoresistenze, joystick eccetera. Sul lato destro dello schema è visibile il nostro consueto circuito di test, per mezzo del quale desideriamo misurare contemporaneamente la tensione ai capi di una resistenza variabile ed il valore della resistenza stessa.
Le misure in questione sono in effetti estremamente semplici da realizzare; dopo avere connesso il circuito di test, composto dalla resistenza R8 da 5,6 KOhm e dal potenziometro R9 da 10 KOhm è sufficiente lanciare il programma ed abilitare la lettura del valore di tensione premendo il pulsante '*Enable Monitoring*' posto sulla form del programma. La lettura del valore di tensione è quindi immediato.
Per quanto riguarda il valore di resistenza del potenziometro, fornisce indirettamente la posizione della manopola. Il valore in questione può essere calcolato come segue, considerando la tensione di alimentazione costante a 5 Volts: *R9=(Vin*R8) /(5-VIn)*. La posizione della manopola semplicemente è: *Posizione (%)=R9/ 10000*100.* Quindi leggendo un valore di tensione pari ad esempio a 2,52 Volt abbiamo:

```
R9=(2,52*5600)/(5-2,52)=5690 Ohm
```

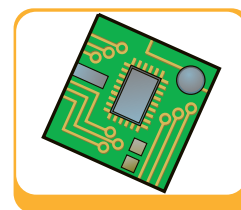
Mentre la posizione della manopola è:

```
Posizione (%)= 5690/10000*100=56,9%
```

Ovvero quasi a metà della sua corsa. Le applicazioni di quanto riportato finora sono innumerevoli ed importantissime e sono alla base di un gran numero di tipologie di controlli industriali.

CONCLUSIONI

Il progetto dello schema elettrico dell'oscilloscopio, tutti i collegamenti necessari, il software compilato ed i relativi codici sorgenti sono stati messi a completa disposizione del lettore. Un doveroso e sentito ringraziamento è dovuto alla Philips Semiconductors per avere permesso la pubblicazione delle informazioni relative al convertitore analogico-digitale ADC804 e ai componenti HEF4011 e HEF4019.
Il lettore vorrà comprendere che nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo e di studio, pertanto l'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dell'utilizzo di quanto esposto in questa sede, soprattutto per la tipologia e la complessità dell'argomento.

Luca Spuntoni

- **ADC803/804 CMOS 8-BIT A/D CONVERTERS, (Philips Semiconductors) 2002.**
- **HEF4019 (Philips Semiconductors) 2002**
- **HEF4019 (Philips Semiconductors) 2003**
- **ESPERIMENTI DI ELETTRONICA DIGITALE PORTE LOGICHE ED OSCILLATORI *Luca Spuntoni* ioProgrammo N. 68 Aprile 2003**

SUL WEB

Il sistema proposto in queste pagine è stato realizzato e collaudato con l'apparecchiatura per il collaudo e la sperimentazione di circuiti elettronici con Personal Computer 'PC EXPLORER light': per ulteriori informazioni su come si possa reperire questa apparecchiatura è possibile visitare il WEB all'indirizzo: www.pcexplorer.it **oppure** http://web.tiscali.it/spuntosoft **od infine inviare una e-mail a:** spuntosoft@tiscali.it

Accesso alla SIM card

I dispositivi smartphone rappresentano una importante novità nello scenario dei dispositivi mobili. Sono molte e interessanti le applicazioni che possono essere sviluppate

In questo articolo descriveremo i componenti software da installare per iniziare lo sviluppo di applicazioni basate su smartphone. Una volta illustrata la procedura di installazione degli strumenti di sviluppo, inizieremo la costruzione di un'applicazione che consentirà l'accesso alla SIM del nostro dispositivo smartphone. Impareremo a costruire le funzionalità che ci permetteranno di accedere alla SIM per leggere i dati in essa memorizzati, come ad esempio i contatti. Vedremo, inoltre, come aggiungere nuovi contatti alla SIM.

LA SIM APPLICATION

Nella prima parte ci occuperemo dell'installazione dei componenti software necessari per lo sviluppo e dell'interfaccia utente dell'applicazione, illustrando il layout della maschere e la navigabilità tra le stesse. Nella seconda parte ci occuperemo di scrivere il codice necessario per accedere alle informazioni memorizzate nella SIM card, illustrando la struttura della classe *WSimManager* che abbiamo realizzato. La classe *WSimManager* è utilizzata già durante la costruzione dell'interfaccia utente, per cui anticipiamo che essa espone due metodi statici fondamentali:

- **GetContatto(..):** permette di ottenere il numero telefonico e la descrizione del contatto in una certa posizione nella lista dei contatti della SIM;
- **InsertNumber(..):** permette di scrivere in una certa posizione nella lista dei contatti, un numero telefonico e una descrizione.

Fig. 1: **Smartphone HTC Qtek8080**

REQUISITI

Conoscenze richieste
C#, .NET Compact Framework

Software
Windows 98 SE /2000/XP, Visual Studio .NET 2003, Active Synch 3.7.1 o superiore

Impegno

Tempo di realizzazione

LA MACCHINA PER LO SVILUPPO

Per la nostra applicazione ci avvarremo del .NET framework. Affinchè le applicazioni sviluppate con l'ambiente di sviluppo Visual Studio .NET 2003 possano "girare" su dispositivi mobili come Pocket PC e smartphone, è necessario che su di essi sia installato una versione "ridotta" del .NET framework: il .NET Compact Framework. E' appena il caso di evidenziare che per dispositivi equipaggiati con il sistema operativo Windows Mobile 2003, sia nella versione per PocketPC che per Smartphone, il .NET Compact Framework è già installato nella ROM. Il componente fondamentale da installare è rappresentato dal SDK per Smartphone 2003. L'installazione dello smartphone 2003 SDK deve essere fatta dopo aver installato, nell'ordine, i seguenti componenti:

Microsoft ActiveSync 3.7.1: il componente smartphone 2003 SDK utilizza questa versione di ActiveSynch per lo sviluppo di applicazioni con Embedded Visual C++ 4.0 e Visual Studio 2003. È necessario eliminare le precedenti versioni di questo software prima di installare la versione 3.7.1. L'operazione di disinstallazione/installazione di ActiveSynch, potrebbe comportare una operazione di "Riparazione" di Visual Studio .NET nel caso esso fosse già installato nel sistema.
eMbedded Visual C++ 4.0 and Service Pack 3: rappresenta il secondo componente da installare nel caso si volessero sviluppare applicazioni direttamente in ambiente Windows CE 4.2.
Visual Studio .NET 2003: rappresenta l'ambiente di sviluppo integrato con cui sviluppare applicazioni con la piattaforma .NET.

Completati i passi precedenti, si può procedere con l'installazione di Smartphone 2003 SDK.

IL PROGETTO

Procediamo ad illustrare passo passo la costruzione del progetto con Visual Studio .NET 2003.

1. Apriamo l'ambiente di sviluppo e nella scheda *"Start Page"* si faccia click sul tasto *"New Project"*.
2. Nella schermata *"New Project"* selezionare *"Visual C# Project"* come tipo di progetto; nel box *"Template"* selezionare *"Smart device Application"*; nella casella di testo *"Name"* specificare *"SimApplication"* per il nome dell'applicazione.
3. Nella schermata *"Smart Device Application Wizard – SimApplication"* si deve specificare nel

primo box *"smartphone"* la piattaforma su cui sviluppare l'applicazione; inoltre, nel secondo box dovremo specificare *"Windows Application"*. (Fig. 2)

***Fig. 2:** L'impostazione della piattaforma "smartphone" quale dispositivo target dell'applicazione*

In Fig. 3 si può osservare come si presenta il nostro ambiente di lavoro al completamento dei passi precedenti. Inoltre, l'ambiente di sviluppo ha creato in automatico la form *"Form1"* che rappresenta la finestra principale della nostra applicazione. Ci proponiamo ora di modificare l'aspetto della nostra form principale.
A questo scopo, dal Toolbox dei controlli di Visual Studio .NET (Fig. 4), abbiamo prelevato e inserito nella maschera una label di benvenuto. Inoltre, sono stati aggiunti due elementi principali di menù: "Nuovo" come elemento attivabile con il soft key sinistro dello smartphone; "Menù" come elemento attivabile con il sofkey destro. In Fig. 5 possiamo osservare l'applicazione eseguita sullo smartphone. Dalla stessa figura si osserva che all'elemento "Menù" sono state aggiunte due voci:

Exit: permette di uscire dall'applicazione *SimApplication*;
Contatti: che permetterà di attivare una maschera in cui visualizzare i contatti memorizzati nella SIM Card dello smartphone.

Dall'illustrazione degli elementi presenti sulla form principale, si osserva la semplicità dell'applicazione in termini di navigabilità. È necessario completare l'insieme delle form con la creazione di una prima form per la visualizzione dei contatti della SIM e di una seconda form per l'inserimento di un nuovo contatto. I passi successivi ci guideranno alla creazione delle due form.

I CONTATTI

In Fig. 6 possiamo osservare la struttura della form *"Contatti"*. Per la sua costruzione, abbiamo aggiunto i seguenti elementi prelevandoli dal Toolbox dell'ambiente di sviluppo:

- Un *MainMenù control,* cui abbiamo inserito la voce *"Fatto"*, la cui pressione con il softkey sinistro dello smartphone ci farà tornare nella maschera principale.
- Un *ListView control,* che sarà popolato dai dati dei contatti letti dalla SIM. Allo stesso controllo abbiamo aggiunto due colonne: *"Contatto"*, che rappresenta la descrizione del mio contatto; *"Numero"*, il relativo numero di telefono.

A questo punto scriviamo il metodo per la gestione dell'evento di selezione della voce di menù principale *"Fatto"*. A tale scopo, visualizzata la form Contatti in *Design View,* occorre fare doppio click sull'elemento *"Fatto";* questa azione ci permetterà la creazione automatica del metodo di gestione di evento, cui dovremo semplicemente aggiungere *this.Close();* l'effetto della precedente istruzione è di chiudere la form attuale e passare alla principale.

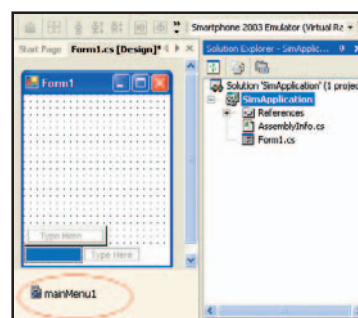



***Fig. 3:** Progetto SimApplication in Visual Studio .NET*

POPOLIAMO LA LISTA DEI CONTATTI

Il popolamento del *ListView control* dei contatti, verrà effettuato durante il caricamento della form *frmListaContatti*. In *design view,* visualizziamo la form *frmListaContatti* e selezioniamo la scheda *Properties* dell'ambiente di sviluppo; a questo punto sarà sufficiente selezionare la scheda degli eventi e fare un doppio click sulla voce *Load*. Sarà così generato tutto il codice necessario per la registrazione dell'evento di load del form e la creazione del template per il relativo metodo di gestione. Ecco il codice che sarà necessario aggiungere al metodo:

```
string nome = string.Empty;
string numero = string.Empty;
string message = string.Empty;
string[] items = new string[2];
for(int i=0; i < WSimManager.TotaleContatti; i++) {
  if(!WSimManager.GetContatto(i,nome,numero,message))
  { break; }
  items[0] = nome;
  items[1] = numero;
  ListViewItem item = new ListViewItem(items);
  this.listView1.Items.Add(item); }
```

La logica di funzionamento è molto semplice: viene utilizzato il metodo static *GetContatto* della classe *WSimManager* che, ricevuto in ingresso l'indice intero i di iterazione, valorizza i campi di tipo string *nome* e *cognome* relativi al contatto i-esimo memorizzato nella SIM card. Per ogni contatto letto, viene aggiunta una nuova riga al *ListView control listView1* della form. L'implementazione del metodo *GetCon-*

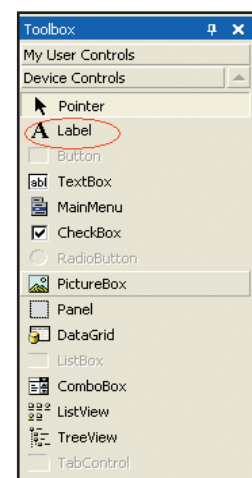



***Fig. 4:** Toolbox dei controlli nell'ambiente di sviluppo*

tatto è rimandata a tra poco. Per completare il lavoro sarà necessario collegare la *frmListaContatti* alla form principale per la navigazione. Per ottenere questo risultato è sufficiente gestire l'evento di selezione dell'elemento di menù *"Lista Contatti"* e aggiungere il codice seguente:

```
frmListaContatti frm = new frmListaContatti();
frm.ShowDialog();
frm.Dispose();
```

Compiliamo ed eseguiamo l'applicazione sullo smartphone premendo la combinazione di tasti *[CTRL]* e *[F5]*. Il codice sarà in automatico scaricato sul dispositivo e mandato in esecuzione. In Fig. 7 possiamo notare la form *Contatti* popolata con i dati della nostra SIM card di test.

Fig. 5: Applicazione in running sullo smartphone

NUOVO CONTATTO NELLA SIM CARD

La voce di menù *"Nuovo"*, nella form principale dell'applicazione, consentirà di attivare una form in cui inserire un nuovo contatto nella SIM. In Fig. 8 possiamo notare in *Design View* l'aspetto della form *frmNuovoContatto* che abbiamo aggiunto al progetto *SimApplication*. Diamo un'occhiata al codice necessario per l'inserimento del nuovo contatto: quello che trovate di seguito deve essere inserito nel metodo di gestione dell'elemento di menù *"Inserisci"* all'interno della form *"New Contatto"*:

```
string nome = txtNomeContatto.Text;
string numero = txtNumero.Text;
string message = string.Empty;
if(!WSimManager.InsertNumber(nome,numero,out
                                        message))
{ MessageBox.Show(message);}
```

Fig. 6: Struttura della form Contatti

Il codice è abbastanza semplice: innanzi tutto vengono caricati dall'interfaccia i dati del nuovo contatto. L'invocazione del metodo static *InsertNumber* della classe *WSimManager* si prenderà cura di inserire i dati dell'utente. L'ultima operazione da fare sulla form *"New Contact"*, consiste nell'aggiungere il metodo di gestione dell'evento di caricamento (evento *Load*). Seguendo la stessa procedura fatta prima per creare in automatico il template del metodo, aggiungiamo ad esso le suguenti linee di codice:

```
txtNumero.Focus();
EditModeHandler.SetNumbersMode();
```

Fig. 7: Popolamento della lista dei contatti letti dalla SIM card

La prima linea di codice ha lo scopo di dare il focus al textbox *txtNumero*, la seconda linea di codice invoca il metodo static *SetNumbersMode* della classe *EditModeHandler*. Questa ultima classe è specializzata nel modificare la modalità di immissione dei dati nei controlli textbox. Infatti, con uno smartphone, agendo sul tastierino, è possibile comporre l'input digitando solo caratteri o solo numeri.

L'ACCESSO ALLA SIM

Per raggiungere lo scopo sarà necessario illustrare il funzionamento di alcune API *(Application Programming Interface)* di Windows CE, fondamentali per l'accesso in lettura e in scrittura ai dati della SIM. La SIM card memorizza diverse tipologie di informazioni: i numeri di telefono e la descrizione dei contatti ad essi associati, i messaggi sms, etc... Un'applicazione per smartphone può accede alla SIM card via codice con l'ausilio di un gruppo di API di Windows CE che costituiscono nel loro insieme quel componente di sistema noto come SIM Manager. Si sottolinea che l'utilizzo di queste API all'interno di un'applicazione sviluppata con Visual Studio .NET 2003 con *.NET Compact Framework*, richiede l'utilizzo della tecnica del *Platform invoke* (o più semplicemente *P/invoke*). Andiamo ad illustrare solo quelle API del SIM Manager di Windows CE che ci permetteranno di gestire i contatti della nostra SIM. Il nostro scopo sarà, dunque, non solo quello di leggere i contatti della SIM e visualizzarli sullo smartphone, ma anche di aggiungerne di nuovi.

CARICAMENTO DEI CONTATTI NELLA SIM

Per accedere a qualunque tipologia di dati memorizzati nella SIM, occorre prima di tutto effettuare un'operazione di inizializzazione della SIM card da codice. L' inizializzazione viene effettuata con la API *SimInitialize* del SIM Manager. Se l'inizializzazione è andata a buon fine, avremo ottenuto un handle valido della nostra SIM che potrà essere utilizzato per l'invocazione delle API di lettura dei dati relativi ai contatti oppure per aggiungere nuovi numeri di telefono, con relativa descrizione, alla SIM. Allo stesso modo, una volta completate le nostre operazioni sulla SIM, è necessario rilasciare il riferimento alla stessa invocando il metodo *SimDeinitialize*. La API *SimReadPhonebookEntry* del SIM Manager permette di accedere a quella parte della memoria SIM che memorizza i numeri di telefono. I parametri che essa accetta sono i seguenti:

- **HSIM hSim:** rappresenta l'handle alla SIM inizializzata invocando l'API *SimInitialize;*
- **DWORD dwLocation:** rappresenta un valore che permette di accedere ad una particolare lista di numeri di telefono; ad esempio, specificando il valore *SIM_PBSTORAGE_OWNNUMBERS* acce-

deremo all'elenco dei nostri numeri personali; specificando il valore *SIM_PBSTORAGE_LASTDIALING* accederemo alla lista delle ultime chiamate effettuate con lo smartphone, etc..

- **DWORD dwIndex:** rappresenta l'indice dell'elemento che vogliamo leggere dalla lista dei contatti;
- **LPSIMPHONEBOOKENTRY lpPhoneBookEntry:** rappresenta un puntatore ad una struttura del SIM Manager nota come *SIMPHONEBOOKENTRY* nella quale verranno memorizzate le informazioni del contatto letto, ovvero, il numero telefonico e la descrizione ad esso associato se disponibile.

A questo punto occorre aggiungere al nostro progetto la classe che permetterà di importare le funzionalità delle API descritte sopra. Sia *WSimManager* la classe aggiunta al progetto. Includere all'inizio di *WSimManager* le seguenti dichiarazioni:

```
[DllImport("cellcore.dll")]
public static extern int SimInitialize(uint dwFlags, int
            lpfnCallBack, uint dwParam, ref int lphSim);
[DllImport("cellcore.dll")]
public static extern int SimDeinitialize(int hSim);
 [DllImport("cellcore.dll")]
public static extern int SimReadPhonebookEntry(int
        hSim, int dwLocation, int dwIndex, ref int
                                  lpPhoneBookEntry);
```

Per completare l'operazione di importazione, è necessario tradurre la struttura *SIM_PHONEBOOKENTRY* in una struttura compatibile con il *.NET Compact Framework*. Questa operazione di traduzione viene chiamata in gergo *"Wrapping"*. Di seguito riportiamo la definizione di *SIM_PHONEBOOKENTRY* in Windows CE:

```
typedef struct simphonebookentry_tag {
   DWORD cbSize;
   DWORD dwParams;
   TCHAR lpszAddress[MAX_LENGTH_ADDRESS];
   DWORD dwAddressType;
   DWORDdwNumPlan;
   TCHAR lpszText[MAX_LENGTH_PHONEBOOKENTRY];
   TCHAR lpszText[MAX_LENGTH_
                        PHONEBOOKENTRYTEXT];
} SIMPHONEBOOKENTRY, *LPSIMPHONEBOOKENTRY
```

Al metodo *SimReadPhonebookEntry* del SIM Manager viene passato un puntatore alla struttura precedente, che sarà caricata con i dati del contatto della rubrica dello smartphone. In particolare, il campo *lpszAddress* della struttura conterrà il numero di telefono del contatto; mentre il campo *lpszText* rappresenta un array di caratteri che mantiene la descrizione associata al contatto. L'operazione di wrapping della struttura precedente consiste

- nel creare una nuova struttura all'interno della classe del progetto;
- far corrispondere ad ogni campo della struttura originaria un relativo campo della nuova struttura in .NET con un tipo che sia compatibile con il precedente.

Il wrapping della struttura *SIMPHONEBOOKENTRY* è abbastanza complicato e la spiegazione della tecnica utilizzata ci farebbe perdere di vista l'obiettivo dell'articolo. Per cui vi rimandiamo alla lettura del codice inserito nella classe *WSimManager* del progetto in allegato; i commenti aiuteranno alla comprensione della logica di wrapping applicata in questo caso. Analizziamo il codice necessario per effettuare la lettura di un generico contatto memorizzato nella SIM. Questa operazione è svolta dal seguente metodo di cui mostriamo il codice:

```
public static bool GetContatto(int index, string nome,
                    string numero, out string message)
{  ...
}
```

Il metodo *GetContatto* effettua, prima di tutto, una inizializzazione di un handle alla SIM *(hSim);* se l'operazione è andata a buon fine, viene creata l'istanza *sPhoneEntry* della struttura *SimPhoneBookEntry.* Questa istanza sarà passata alla funzione *SimPhoneBookEntry(..)* per leggere un numero dalla rubrica. Se l'operazione di lettura va a buon fine vengono valorizzati i parametri di ingresso nome e numero che saranno utilizzati dal metodo della classe chiamante. Inoltre, il primo parametro di ingresso index, di tipo intero, rappresenta la posizione che il contatto occupa nella lista della rubrica della SIM.

INSERIMENTO DI UN NUOVO CONTATTO

Il metodo statico *InsertNumber()* della classe *WSimManager,* permette l'inserimento di un numero di telefono con relativa descrizione nella memoria della SIM. La differenza sostanziale rispetto al metodo *GetContatto(..)* precedente, è rappresentato dal fatto che i parametri nome e numero di tipo string sono parametri di ingresso e non di uscita. L'implementazione completa la trovate nel codice allegato:

```
public static bool InsertNumber(…)
```

Ora siamo pronti per sviluppare altre applicazioni che possano potenziare il nostro smartphone!

Ing. Elmiro Tavolaro

Fig. 8: **Form di inserimento di un nuovo contatto**

SUL WEB

È possibile prelevare Visual C++ 4.0 com SP3 al seguente indirrizzo:
http://www.microsoft.com/downloads/details.aspx?familyid=5bb36f3e-5b3d-419a-9610-2fe53815ae3b&displaylang=en

È possibile prelevare ActiveSync 3.7.1 al seguente indirizzo:
http://www.microsoft.com/windowsmobile/downloads/activesync37.mspx

È possibile prelevare Smartphone 2003 SDK al seguente indirizzo:
http://www.microsoft.com/downloads/details.aspx?FamilyID=a6c4f799-ec5c-427c-807c-4c0f96765a81&displaylang=en

Siti WEB facilmente aggiornabili e visibili su cellulari i-mode

Gestione remota di siti WEB

parte prima

Costruiremo un sito in cui sia possibile effettuare l'aggiornamento dei contenuti da remoto, introdurremo anche una sezione dedicata ai servizi i-mode

☑ CD ☑ WEB
IFP.zip
cdrom.ioprogrammo.it

Analizziamo un possibile scenario: un istituto di formazione professionale che organizza corsi di informatica, vuole inserire nel suo sito una sezione per fornire agli studenti iscritti ai corsi alcuni servizi: consultazione di articoli redatti dagli insegnanti, prenotazione esami e consultazione dei corsi attivati. L'istituto desidera inoltre fornire ai docenti sezioni a loro dedicate e da loro gestite e aggiornate da remoto: ogni docente potrà inserire articoli, inviare comunicazioni ai suoi studenti e visionare gli articoli presenti sul sito, dal computer del suo studio o da casa. Anche l'amministratore (potrebbe essere anche lui un docente), che provvederà alla gestione degli utenti, sia studenti che docenti, e dei corsi, deve poter operare in remoto. Vi deve essere anche una sezione pubblica consultabile da tutti da cui è possibile vedere i corsi attivati e i programmi relativi ad ogni corso.

Fig. 1: **Diagramma delle relazioni tra le tabelle**

LA BASE DI DATI

Il database utilizzato nell'applicazione è stato creato con *ACCESS* e contiene le seguenti tabelle: *Corsi, Docenti, Studenti, Articoli, StudentiIscritti, Prenotazioni.* Dopo aver preparato il Database, si deve creare, tramite *Origini Dati ODBC* contenuto nel pannello di controllo, un DSN *(Data Source Name)* di sistema per:

- assegnare un nome logico alla sorgente di dati;
- specificare il driver da utilizzare;
- specificare il percorso fisico del database.

Un DSN permette di astrarci dalla posizione fisica del database che può così essere spostato in un'altra cartella senza apportare nessuna modifica al codice già scritto; basterà modificare il percorso nel DSN di sistema. Il nome per l'origine dati utilizzato per questo database è *IstitutoIFP.*

STRUTTURA DEL SITO

Per entrare nelle aree riservate sarà necessario che gli utenti effettuino il *logIn*. Dalla homepage (file *index.asp*) si potrà accedere al Form di login che richiamerà la pagina *Esegui_Login* in cui vengono eseguiti i controlli su username e password inseriti:

```
set IFPDB=server.createObject("ADODB.Connection")
IFPDB.open "IstitutoIFP"
sqlDoc="select * from docenti where username=
                              '" & p_usernameDoc & "'"
...
IFPDB.close
  set IFPDB=Nothing
response.redirect "/ifp/index.asp"
```

Se l'utente è registrato, viene creato un cookies in cui vengono salvati *Cognome, Nome, TipoUtente* (studente, docente, admin), *IDutente*. Per effettuare il logout basterà impostare come data di scadenza del cookie una data anteriore alla data attuale: *response. cookies("isLogged").expires="#01/01/2000 00:00: 00#.* Per accedere al database viene utilizzato un oggetto ADO *(ActiveX Data Object)*. Questi oggetti consentono di collegarsi ad un database per compiere le operazioni di inserimento, aggiornamento, modifica, ricerca, cancellazione. Sono oggetti COM che utilizzano l'interfaccia di programmazione ODBC *(Open Data Base Connectivity)* per accedere ai dati. Il file *indexRiga.txt* incluso in ogni pagina del sito con una operazione di *Server Side Include,* contiene l'intesta-

zione delle pagine: il titolo e i collegamenti a tutte le sezioni. Prima di inviare la pagina al browser, il server controlla se è stato già effettuato il login; se sì, nella pagina appare il collegamento alla pagina di logout con il nome e il tipo utente:

```
<%p_username=request.cookies("isLogged")
  p_cognome=request.cookies("isLogged")("Cognome")
  p_nome=request.cookies("isLogged")("Nome")
  p_TipoUtente=request.cookies("isLogged")("TipoUtente")
%>
.
<%  if p_username="" then%>
   <A href="/ifp/form_login.asp" > Login </A>
    
<% else %>
   <A href="/ifp/form_logout.asp" > LogOut
    >> 
   <%=p_cognome%>  &#32 <%=p_nome%> (
                                <%=p_TipoUtente%> )</a>
<% end if%>
```

L'area pubblica

L'unica area del sito accessibile agli utenti non registrati è la sezione in cui è possibile vedere i corsi attivati dall'Istituto di Formazione. Interrogando il database è possibile estrarre tutte le informazioni relative ai corsi e il docente per ogni corso. Il programma del corso, invece, è contenuto in un file asp separato, ma collegato ad un campo della tabella visualizzata nella pagina (il suo nome è memorizzato in un campo della tabella *Corsi*). Per accedere al database, si deve creare un'istanza di tipo *ADODB. Connection* per la connessione al database e utilizzare un'istanza di tipo *ADODB.RecordSet* per estrarre le righe della tabella. Viene incluso in molte pagine il file *adovbs.inc* che contiene le costanti VbScript per la gestione degli oggetti ADO.

```
set IFPDB=Server.CreateObject("ADODB.Connection")
IFPDB.Open "IstitutoIFP"
set tableSet=Server.CreateObject("ADODB.Recordset")
tableSet.Open "select * from Corsi order by
                                        DataEsame", IFPDB
```

Se non sappiamo di quanti campi è composta la tabella, possiamo ricavare questa informazione dalla proprietà *Fields.Count*:

```
p_numerOfColumns=tableSet.Fields.Count.
```

Col ciclo seguente si potranno estrarre tutte le righe dalla tabella:

```
<% While not tableSet.EOF %>
<TR>
<%for x = 1 to (p_numerOfColumns-3)
  if x=4  and tableSet.Fields(x).value<>"" then %>
    <TH> <a href=<%=Server.URLEncode(
                        Cstr(tableSet.Fields(x).value))%>>
      Programma <%=Cstr(
                   tableSet.Fields(1).value)%></a></TH>
  <%else%>
    <TH> <%=tableSet.Fields(x).value %></TH>
  <%end if
next
..
<% tableSet.MoveNext
wend
```

Il metodo *Server.URLEncode* (stringa) traduce in un indirizzo valido il contenuto della stringa, in quanto in un URL alcuni caratteri non sono consentiti: se, per esempio, nella stringa sono contenuti degli spazi saranno rimpiazzati con i caratteri +. Per estrarre il nome del docente si utilizza il suo ID prelevato dalla tabella corsi. In questo caso è stato utilizzato il metodo *Execute* dell'oggetto *ADODB.Connection* (il cui valore di ritorno è comunque un RecordSet) per estrarre i dati dalla tabella *Docenti:*

Fig. 2: HomePage del sito

```
sqlText=""
sqlText="select * from docenti where IDdocente=
                                  " & Cstr(id_docente)
set userSetDoc=IFPDB.Execute(sqlText)
if (not userSetDoc.EOF) then
   NomeDocente=trim(userSetDoc.fields.item("Cognome"))
   NomeDocente=NomeDocente & " " & trim(
                        userSetDoc.fields.item("Nome"))
else NomeDocente=""
end if set userSetDoc=nothing
```

Area studenti

Gli studenti che si iscrivono ai corsi, ricevono uno username e una password che consente loro di visitare le pagine riservate. Lo studente può visitare la sezione articoli del corso/i a cui è iscritto e può inviare una richiesta di prenotazione per un esame(riceverà una e-mail di conferma se la prenotazione è andata a buon fine).

Visione degli articoli

Dalla pagina *StudentIndex.asp* ci si collega alla pagina *formArticoliStu.asp,* contenuta nella cartella *ifp \Docenti\Articoli.* Viene controllato che l'utente sia loggato andando a prelevare le informazioni contenute nel cookie e viene interrogato il database con l'istruzione SQL

```
SELECT Studenti.IDstudente, Studenti.UserName,
   StudentiIscritti.IDcorso, StudentiIscritti.CodiceCorso
        FROM Studenti INNER JOIN StudentiIscritti ON
```

ODBC

Strato software che fa da interfaccia tra la fonte di dati e le applicazioni, rendendole di fatto indipendenti.

STRUMENTI NECESSARI

- **ASP per la gestione di pagine WEB dinamiche**
- **ACCESS, SQL e ADO per la gestione del database**
- **HTLM**
- **PWS (Personal Web Server) o IIS (Internet Information Services) per testare in locale le nostre pagine ASP**
- **Componenti per inviare e-mail dalla pagina ASP e per effettuare l'upload**
- **cHTLM**
- **Un simulatore per testare le pagine per i-mode.**

Codice	Titolo	Descrizione	Programma del corso	Data esame	Docente
ECDLMO5	ECDL Modulo 5 ACCESS	Corso di preparazione all'esame ECDL Modulo 5 relativo ad ACCESS	Programma ECDLMO5	10/10/02	Bianchi Bianca
ECDLMO3	ECDL Modulo 3 WORD	Corso di preparazione all'esame ECDL modulo 3: Word	Programma ECDLMO3	12/06/04	Martemucci Patrizia
ECDLMO4	ECDL Modulo 4 EXCEL	Corso di preparazione all'esame ECDL modulo 4: Excel	Programma ECDLMO4	30/06/04	Rossi Andrea
CBUILDER	Programmazione in	Corso di programmazione a oggetti in C++. Ambiente di sviluppo	Programma	24/02/05	Martemucci

Fig. 3: **La pagina che visualizza l'indice dei corsi attivati**

SUL WEB

Il software Persits è scaricabile liberamente all'indirizzo www.aspemail.com/download.html Dopo averlo installato, bisogna registrare le due DLL *AspEmail.dll* e *AspUpload.dll* col comando Regsvr32.

NOTA

COMPONENTI UTILI

Per questa applicazione è stato utilizzato il componente AspEmail dal software Persits, scaricabile liberamente dal sito www.aspemail.com/download.html. Persits comprende anche una copia di prova valida trenta giorni del componente *AspUpload.* Installato il componente, sarà necessario registrare le dll eseguendo i comandi:

```
regsvr32 c:\Programmi\Persits Software\AspEmail\BIN\AspEmail.dll
regsvr32 c:\Programmi\Persits Software\AspEmail\BIN \AspUpload.dll
```

```
Studenti.IDstudente =
StudentiIscritti.IDstudenteIscritto WHERE
((Studenti.IDstudente= " & p_IDutente &"))
```

Tramite l'ID dello studente vengono prelevate le informazioni dalle tabelle *"Studenti"* e *"StudentiIscritti"* per visualizzare i corsi ai quali lo studente è iscritto. L'istruzione *SQL JOIN* il risultato recupera un recordset contenente le righe della prima e della seconda tabella che hanno valori uguali dell'attributo comune *(IDstudente)* e che soddisfano il criterio di selezione *(clausola where)*. Il codice seguente serve per elencare i corsi e creare i collegamenti alla pagina asp *Esegui_ FormArticoliStu.asp* a cui viene passato come parametro il codice del corso:

```
<h4 style="color:rgb(0,0,240)"> Corsi ai quali sei
iscritto</h4>
<P>
<TABLE BORDER=0>
<%While not tableSet.EOF %>
<TD>
<A HREF="Esegui_FormArticoliStu.asp? p_CodiceCorso
   =<%Response.Write tableSet("CodiceCorso")%>">
  <%=tableSet.Fields(3).value%>
  </A></TD>
</td></tr>
<% tableSet.MoveNext
  wend
  tableSet.Close
  set tableSet=Nothing
  IFPDB.Close
  set IFPDB=Nothing %>
</TABLE>
```

Nella pagina *Esegui_FormArticoliStu*, dopo tutti i controlli vari, viene eseguita la query

```
SELECT [Articoli].[CodiceCorso], [Docenti].[Cognome],
[Docenti].[Nome], [Articoli].[TitoloArticolo],
[Articoli].[DataPubblicazione], [Articoli].[IndirizzoFile]
FROM Docenti INNER JOIN Articoli ON
[Docenti].[IDdocente]=[Articoli].[IDdocente] GROUP
BY [Articoli].[CodiceCorso], [Docenti].[Cognome],
[Docenti].[Nome], [Articoli].[TitoloArticolo],
[Articoli].[DataPubblicazione], [Articoli].[IndirizzoFile]
HAVING (((Articoli.CodiceCorso)= '" & p_CodiceCorso &
"')) ORDER BY [Articoli].[DataPubblicazione] DESC
```

che preleva tutti gli articoli del corso che soddisfano la clausola *Having*. Con un ciclo *While* vengono poi visualizzati in una tabella che conterrà per ogni articolo il collegamento al file corrispondente:

```
<% While not tableSet.EOF
p_NomeDocente=cstr(tableSet.Fields(1).value) & "
" & cstr(tableSet.Fields(2)) %>
<TR><TH><%=cstr(tableSet.Fields(0).value)%></TH>
<TH><%=p_NomeDocente%></TH>
<%for x = 3 to (p_numerOfColumns-2)
  if x=3  then %>
    <TH> <a href=<%=Server.URLEncode(Cstr(
tableSet.Fields(5).value))%>>
    <%=Cstr(tableSet.Fields(3).value )%></a></TH>
    <%else%>
    <TH> <%=tableSet.Fields(x).value %></TH>
  <%end if
next
tableSet.MoveNext
wend
```

Prenotazione Esami

La pagina *FormPrenotazioni.asp* elenca i corsi ai quali uno studente è iscritto e collega il corso alla pagina *Esegui_FormPrenotazioni.asp* a cui passa come parametro il codice del corso. Quest'ultima controlla che la prenotazione non sia già stata effettuata e inserisce i dati nella tabella prenotazioni:

```
' controlla che la prenotazione non sia già registrata
tableSet.open "SELECT * FROM Prenotazioni WHERE(((
Prenotazioni.IDcorso)=" & p_IDcorso AND
((Prenotazioni.IDstudente)=" & p_IDutente & "))", IFPDB
if tableSet.EOF then 'puoi inserire i dati
set inserPre=Server.CreateObject("ADODB.RecordSet")
inserPre.Open "Prenotazioni",IFPDB, adOpenDynamic,
adLockOptimistic, adCmdTable
inserPre.AddNew
inserPre("CodiceCorso")=p_CodiceCorso
inserPre("IDcorso")=Cint(p_IDcorso)
inserPre("IDstudente")= Cint(p_IDutente)
inserPre("DataPrenotazione")= Date
inserPre.Update
inserPre.close
set inserPre=nothing %>
<h2>Prenotazione effettuata.</h2><br>
<h2>Riceverai una email di conferma.</h2>
```

A questo punto interviene il componente *Persits* che ci permette di inviare email dalla pagina asp. Si deve creare una instanza dell'oggetto *Persits.MailSender*

```
Set Mail = Server.CreateObject("Persits.MailSender")
Mail.Host="SMTP.tiscali.it"
```

e preparare tutti i parametri per l'invio

```
Mail.Host="SMTP.tiscali.it" 'nome server SMTP
Mail.AddAddress p_email
Mail.From="IndirizzoScuola@tiscali.it"  'indirizzo del
mittente
Mail.FromName="Istituto di formazione professionale"
Mail.Subject="Conferma prenotazione"
```

```
p_body="Comunicazione corso " & p_codiceCorso
p_body=p_body & chr(13) & chr(10) & chr(13) & chr(10)
p_body=p_body & "La tua prenotazione è stata
                    registrata. " & chr(13) & chr(10)
p_body=p_body & "Data Esame: " & cstr(
                    p_DataEsame) & chr(13) & chr(10)
p_body=p_body & p_testo
Mail.Body=p_body
on error resume next
  if not Mail.Send then
    response.write "Problemi nella trasmissione della
                                        email  <BR>"
  end if
on error goto 0
```

Inserimento dati utente

I dati di ogni utente vengono raccolti nel form *InserimentoUtenti.asp* collegato alla pagina *Esegui_Inserimento_Utente.asp*. Fatti gli opportuni controlli, i dati vengono inseriti nel database:

```
set inserUtente=Server.CreateObject("ADODB.RecordSet")
inserUtente.Open "Docenti",IFPDB, adOpenDynamic,
                        adLockOptimistic, adCmdTable
    inserUtente.AddNew
    inserUtente("Cognome")=p_Cognome
    inserUtente("Nome")=p_nome
    inserUtente("UserName")= p_UserName
    inserUtente("Password")= p_Password1
    inserUtente("Gruppo")= p_tipo
    inserUtente("email")= p_email
  inserUtente.Update
  inserUtente.close
  set inserUtente=nothing
```

Modificare i dati di un corso

Di un corso è possibile, una volta inserito, cambiare solo i dati relativi al docente e la data dell'esame oppure cancellarlo completamente. Si accede tramite il link contenuto in *DocentiIndex.asp* alla pagina *ModificaCorsi.asp:*

```
<% set tableSet=Server.CreateObject(
                                "ADODB.RecordSet")
tableSet.Open "select * from Corsi order by IDcorso",
        IFPDB, adOpenForwardOnly, adLockOptimistic,
                                        adCmdText
…
</TABLE>
<h4>Non è possibile cancellare un corso con la data di
            esame maggiore della data attuale! </h4>
<INPUT TYPE="submit" VALUE="Cancella i corsi selezionati">
</FORM>
```

Il primo campo della tabella è un CheckBox che, se selezionato, assume il valore di *IDcorso* estratto dalla tabella *Corsi*. Il secondo campo contiene un collegamento alla pagina *Modifica_Corsi_ Rec.asp* a cui viene passato come parametro il valore *IDcorso*. Se si clicca sul pulsante *Cancella i corsi selezionati* viene richiamata la pagina *Modifica_Corsi_Esegui.asp* che provvederà a cancellare i corsi dalla tabella e ad eliminare i file contenenti il programma del corso selezionato. La procedura è la stessa che viene utilizzata per la cancellazione degli articoli:

```
if request.cookies("isLogged")("TipoUtente")="admin" then
  set fsObject=Server.CreateObject(
                        "Scripting.FileSystemObject")
  set NomeFileCanc=Server.CreateObject(
                        "Scripting.FileSystemObject")
  p_cartella="c:\inetpub\wwwroot\ifp\cestino"
  set folderobj=fsObject.getFolder(p_cartella)
    for each  fileObject in folderobj.Files
        p_file=fileObject.name
        p_full_path=(p_cartella & "\" & p_file)
        NomeFileCanc.deleteFile(p_full_path)
   next
  set fsObject=nothing
  set folderobj=nothing
```

Se invece si clicca sul campo *IDcorso*, viene richiamata la pagina *Modifica_Corsi_rec.asp*, a cui viene passato anche l'*IDcorso*, e che contiene un form in cui vengono elencate le informazioni del corso che non possono essere modificate e quelle che possono essere modificate. Da qui si passa alla pagina *Modifica_Corsi_Rec_esegui.asp* che aggiorna dei dati nella tabella *corsi*:

Fig. 4: *Form per la raccolta dei dati relativi ad un utente*

```
set insertSet=Server.CreateObject("ADODB.RecordSet")
insertSet.Open "Corsi",IFPDB, adOpenDynamic,
                        adLockOptimistic, adCmdTable
insertSet.Find "IDcorso= " & p_link_id
if err.number=0 then
        insertSet("DataEsame")=p_link_data
        insertSet("IDdocente")=p_link_IDdocente
        insertSet.Update
        insertSet.Close
        set insertSet=Nothing
IFPDB.Close
        set outpostDB=Nothing
end if
```

CONCLUSIONI

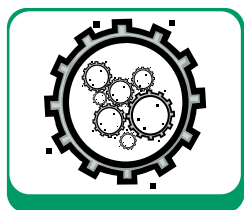
Siamo ormai a metà strada nella costruzione del nostro sito: abbiamo visto come gestire gli utenti e i corsi dalla parte *Amministratore* e abbiamo visto come presentare alcuni servizi agli utenti-studenti.

Patrizia Martemucci

Se si ha qualche difficoltà nell'impostare interrogazioni SQL, si può far generare la query ad ACCESS seguendo la procedura guidata. La pagina Visualizza/SQL contiene il codice generato da ACCESS. Dopo averlo copiato nella pagina ASP, basterà solo ritoccarlo leggermente inserendo le variabili vbscript al posto dei parametri impostati da ACCESS.

Cosa si cela dietro la prossima rivoluzione tecnologica

Tag RFID visti da vicino

Nei due precedenti articoli abbiamo visto come realizzare un gestionale di magazzino sfruttando la tecnologia dei tag a radiofrequenza. Vediamo ora cosa succederà con i loro diretti successori: gli RFID

Nei precedenti articoli abbiamo visto come utilizzare i tag a radiofrequenza per etichettare dei prodotti in magazzino, e come questa tecnologia, se opportunamente utilizzata, riduca notevolmente i tempi di lavorazione e, di conseguenza, i costi. Spesso non ci facciamo caso ma, inconsapevolmente, siamo già circondati da questi dispositivi e tale tecnologia è già ampiamente utilizzata. Non ci credete? Pensate ad esempio al Telepass: è un tag attivo! Se avete un animale domestico, con tutta probabilità gli sarà stato impiantato un "chip" di riconoscimento: anche quello è un tag a radiofrequenza. E l'elenco potrebbe continuare a lungo! Come tutte le tecnologie però, anche i nostri tag si stanno evolvendo e, il loro successore si chiama RFID. Lo scopo primario della tecnologia RFID è quello di poter catalogare in modo univoco ogni singolo prodotto. Dalla fabbrica alla rivendita finale, ogni prodotto potrà sempre essere rintracciato e, automaticamente, si potranno avere tutte le informazioni relative ad esso. Catalogazione ed inventario saranno solo ricordi del passato. Quando un camion esce dalla fabbrica, viene automaticamente aggiornato il magazzino. Stesso discorso per tutti i passaggi che porteranno il prodotto fino al cliente finale. Ma come è possibile tutto ciò? Scopriamolo!

LA TECNOLOGIA RFID

Radio Frequency IDentification: questa è la parola chiave alla base di questa tecnologia che promette di rivoluzionare molti settori, soprattutto quelli della logistica e della distribuzione. Ma vediamo di cosa si tratta, facendo dei paragoni con gli attuali tag.
Innanzitutto ritengo indispensabile precisare che questa tecnologia è ancora in fase di sviluppo. Se ne sente già parlare molto, esistono delle specifiche e degli enti che le rilasciano, ma molti dettagli continuano a cambiare. Da quando ho iniziato ad interessarmene, diverse cose sono già state modificate, qualcosa è anche stata eliminata. Non stupitevi quindi se, effettuando delle ricerche su Internet (magari anche nei siti segnalati in questo stesso articolo), troverete delle differenze rispetto a quanto scritto in queste pagine. La cosa importante è comprendere cosa c'è alla base e cosa ci si può fare. I dettagli implementativi si potranno analizzare quando sarà stato tutto definito. Tecnologicamente parlando, un tag RFID non ha nulla di diverso da quelli già visti nel primo articolo di questa serie. Si tratta sempre di un piccolissimo trasmettitore che viene alimentato da un lettore specifico attraverso la sua antenna. La vera rivoluzione sta nella gestione del codice che esso trasmette. Nei tag attualmente in commercio, il codice in essi memorizzato non trasporta nessuna informazione. È un numero seriale che tocca a noi gestire con i nostri sistemi. È evidente che la sua utilità è confinata all'interno della nostra azienda e dei nostri magazzini. Lo scopo che si intende raggiungere con la tecnologia RFID è quello di estendere l'utilità del codice contenuto nel tag anche all'esterno delle singole strutture. Il primo passo è quindi quello di fare in modo che il codice stesso "trasporti" delle informazioni. Il codice si chiama EPC *(Electronic Product Code)* ed è rilasciato da un ente internazionale chiamato *EPCglobal*.

Fig. 1: *Il codice EPC a 96 bit*

Come visibile in Fig. 1, il codice da 96 bit è diviso in 4 sezioni:

- **Intestazione**: è un codice di 8 bit che specifica la versione dell'EPC utilizzato. Al momento ne esiste una sola.

- **EPC Manager**: è un codice di 28 bit che identifica in modo univoco una azienda.

- **Object Class**: è un codice di 24 bit che identifica la categoria di prodotto (bevande lisce, bevande gassate – per riprendere gli esempi del precedente articolo)

- **Serial Number**: identifica in modo univoco un prodotto. Il codice è di 36 bit.

Il codice di 96 bit consente di catalogare 268 milioni di aziende, 16 milioni di categorie di prodotti per azienda e 68 miliardi di prodotti. L'enorme quantità di elementi catalogabili (già ampliata dal codice EPC di 256 bit), consente di etichettare in modo univoco ogni singolo prodotto. Il vantaggio maggiore che ne deriva è che ogni prodotto sarà sempre riconoscibile, sia che si trovi nella nostra azienda, sia che si trovi, ad esempio, in un centro di distribuzione.

LA GESTIONE DELLE INFORMAZIONI

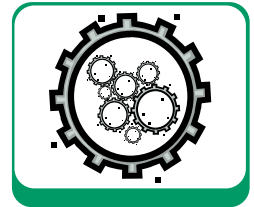
Nel precedente paragrafo abbiamo analizzato l'unità base della tecnologia RFID: il codice EPC. Le informazioni trasportate dal codice dovranno però essere innanzitutto estratte e successivamente gestite. Ed è questo il punto dell'architettura RFID più critico. Le criticità maggiori sono relative a due processi:

- La quantità di informazioni da gestire.

- La gestione dei dati associati ai tag.

È evidente che una grossa azienda che dovesse decidere di utilizzare questa tecnologia, si troverà a dover leggere enormi quantità di dati in pochi secondi. Il rischio di perdita di dati o di saturazione del canale di trasmissione è quindi elevato. Per far fronte a queste problematiche, l'EPCglobal ha rilasciato delle specifiche di un middleware inizialmente denominato *Savant,* ma che ora si chiama ALE *(Application Level Events).* Tale componente si interpone tra i lettori di tag e i sistemi di gestione aziendale ed ha lo scopo di coordinare la lettura e l'invio delle informazioni ricevute. Il middleware, oltre che a coordinare i vari lettori, recupera i dati relativi ai prodotti in base al codice letto. Tali dati sono memorizzati in appositi server denominati PML, e recuperati grazie a ONS *(Object Naming Services).* Come il DNS *(Domain Name System)* per Internet, l'ONS mappa il codice letto dal lettore sulle informazioni presenti sul server PML. PML *(Product Markup Language)* è una implementazione specifica di XML che descrive il prodotto selezionato. In termini strettamente pratici, quando un lettore legge un tag RFID, il middleware ALE (ex *Savant*) utilizza un servizio ONS per recuperare i dati del prodotto sotto forma di PML. Tali dati possono essere gestiti nei classici gestionali già presenti in azienda. Nell'ottica della distribuzione è evidente che i dati espressi in forma PML, non possono essere archiviati all'interno dell'azienda ma devono essere accessibili anche dall'esterno. Solo così il centro di distribuzione (o qualsiasi altro soggetto della catena), potrà ricevere le suddette informazioni. Di questo se ne occupa Verisign che ha messo a disposizione la sua struttura per realizzare un EPC Network.

IL CICLO DI VITA

Seguiamo passo passo il ciclo di vita di un prodotto per comprendere appieno i vantaggi della tecnologia RFID. Quando un prodotto viene realizzato, verrà applicato ad esso un tag RFID che conterrà un codice univoco e standardizzato. Abbiamo visto nel dettaglio il codice EPC nel primo paragrafo. Le informazioni relative a questo prodotto, come la data di scadenza, la temperatura di conservazione ecc., verranno tradotte in formato PML ed archiviare su dei server raggiungibili via internet. Verisign si occupa della gestione dei suddetti server. Visto l'enorme numero di informazioni che viaggeranno sulla rete, è stato previsto un sistema di caching dei dati all'interno delle aziende. ALE, il middleware, oltre ad interfacciarsi con i lettori, si interfaccerà con i server ONS che renderanno possibile il recupero delle informazioni espresse in PML dagli appositi server.
Sarà lo stesso ALE ad archiviare su di essi i dati relativi ai nuovi prodotti. Quando i prodotti usciranno dal magazzino del produttore, un lettore da varco si occuperà di aggiornare sia il magazzino (sempre attraverso il middleware ALE), sia a comunicare ai server PML che il prodotto è uscito dal magazzino del produttore, specificandone anche il luogo di

SUL WEB

Tutte le specifiche ufficiali relative alla tecnologia RFID possono liberamente essere scaricate dal sito dell'EPCglobal e precisamente dall'indirizzo
http://www.epcglobalinc.org/standards_technology/specifications.html

Sul sito dell'EPCglobal è disponibile un tool per il calcolo dei costi di implementazione. I risultati di tale tool devono ritenersi comunque solo a scopo informativo. Il link diretto al tool è il seguente:
http://archive.epcglobalinc.org/howtoadopt_roi.asp

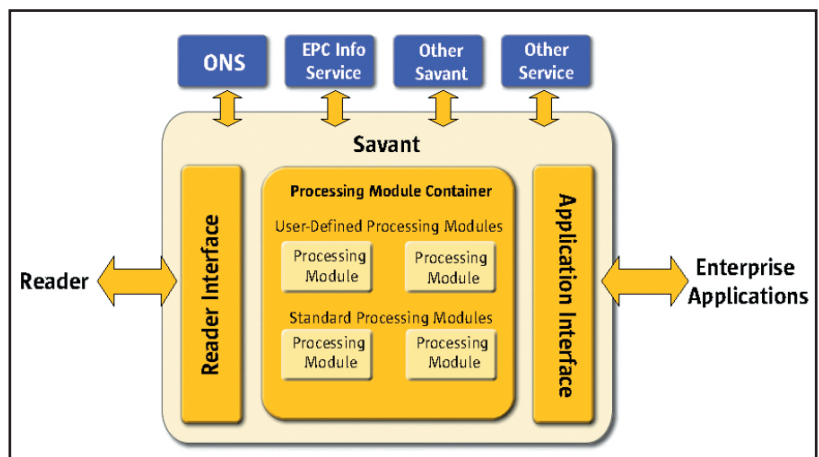
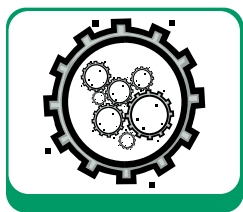



Fig. 2: ***Schema a blocchi del Savant***

destinazione. Il percorso contrario si verificherà quando il carico arriverà in un centro di distribuzione. Un lettore da varco comunicherà al middleware ALE del centro di distribuzione il carico ricevuto, le cui informazioni, una volta lette dai server PML, verranno utilizzare per aggiornare il magazzino del distributore. In questo modo sarà sempre possibile rintracciare ogni singolo prodotto durante tutto il suo ciclo di vita. In futuro potrà essere possibile utilizzare le stesse informazioni per fare una lista della spesa automatica, grazie a dei lettori installati direttamente in casa, o ancora automatizzare la selezione dei rifiuti in base al loro tipo.

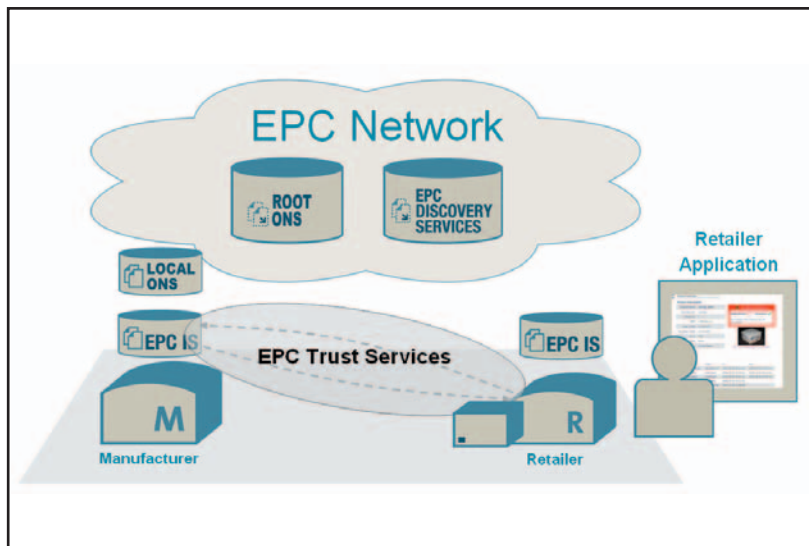



Fig. 3: **Uno schema dell'EPC Network messo a punto da Verisign**

I NOSTRI GESTIONALI

Ora che la tecnologia degli RFID è un po' più chiara, analizziamo gli aspetti che riguardano il lato che più ci interessa: lo sviluppo di applicativi.
È fuor di dubbio che questa tecnologia rivoluzionerà tanto i processi produttivi e di distribuzione dei prodotti, quanto la struttura dei software gestionali. Fortunatamente, l'*EPCglobal* ha previsto che, nella maggior parte dei casi, chi utilizzerà questa tecnologia ha già dei sistemi gestionali costati anche parecchio. Se riprendiamo la Fig. 2 vedremo infatti che sul lato destro è presente un blocco denominato *"Application Interface"* il cui scopo è quello di comunicare con le applicazioni aziendali.
I produttori non dovranno fare altro che realizzare gli opportuni moduli per la gestione delle informazioni ricevute dal middleware.
In caso di nuove realizzazioni il discorso cambia in quanto ci sono delle procedure che non serviranno più (vedi il conteggio dei prodotti ad esempio), e dei processi che cambieranno radicalmente come ad esempio l'inventario visto nel precedente articolo.

L'*EPCglobal* è l'ente internazionale che si occupa della standardizzazione della tecnologia RFID. Tutte le info relative all'ente e a come entrare a farne parte, si trovano sul sito ufficiale www.epcglobalinc.org

Si parte innanzitutto dalla gestione dei dati relativi ai prodotti che, come abbiamo visto, arriveranno dal middleware in formato XML (PML per la precisione).
Per la loro gestione dovremo prevedere un modulo software che si occupi di "tradurre" il PML in dati leggibili e, magari, di archiviarli in un nostro Data Base. A questo scopo ci può venire in contro il .net framework che contiene già delle classi per la gestione dei file XML da cui possiamo direttamente creare una struttura dati (un Dataset ad esempio). Dal Dataset al Data Base poi, il passo è estremamente semplice.
Verrebbe da chiedersi il perché archiviare comunque i dati dei prodotti, visto che essi saranno sempre e comunque recuperabili. Anche se di primo impatto potrebbe sembrare una triplicazione di dati (le specifice EPC prevedono anche una cache locale dei dati PML), meglio mettersi al sicuro. Le informazioni che circoleranno sulla nostra rete saranno enormi, quindi, se possiamo in qualche modo ridurle, è tanto di guadagnato. In secondo luogo sarebbe assurdo dover andare a leggere un tag solo per recuperare i dati caratteristici di un prodotto specifico. Pensiamo inoltre alle funzionalità di reportistica di cui avremmo sicuramente bisogno, nonché alla possibilità di arricchire i dati standard delle specifiche PML con dei nostri dati personalizzati (ad esempio la posizione di magazzino, o il responsabile di una determinata categoria di merci).
Non dovremo invece occuparci più della catalogazione dei prodotti stessi. Nel software analizzato il mese scorso, dovevamo inserire prima i dati relativi ai prodotti, poi etichettarli, ed in seguito associare ogni tag ad un prodotto. Con la tecnologia RFID, invece, nel momento stesso in cui i prodotti arriveranno nel nostro magazzino, verranno automaticamente catalogati.
Il discorso è leggermente più complesso se visto dal lato della produzione ma, almeno ad oggi, non ci sono abbastanza informazioni a riguardo.

A CHI CONVIENE

Come tutte le tecnologie, anche quella degli RFID ha sia lati positivi che lati negativi.
Abbiamo finora visto quanto di bello e di comodo ci offre questa tecnologia nascente. Vediamo ora quando conviene utilizzarla e quando è meglio continuare ad utilizzare gli attuali tag.
Innanzitutto dobbiamo tenere in considerazione i costi. Non è una tecnologia gratuita quindi la sua implementazione avrà un costo, tanto per l'acquisto dell'hardware quanto per l'adeguamento delle strutture. Senza ovviamente dimenticare gli eventuali costi dell'affiliazione al circuito EPCglobal. Della tecnologia RFID sono pubbliche le specifiche,

quindi ogni produttore software potrà implementare le sue soluzioni. Ma non dimentichiamo che c'è un Ente che dovrà assegnare e distribuire i codici, e Verisign che darà accesso al network EPC. Ad oggi non ci sono informazioni in merito a questi costi, quindi non è possibile effettuare delle stime. Ma ci sono pochi dubbi che l'adesione al network avrà un costo. È anche molto difficile effettuare delle stime relative ai costi dell'Hardware. La tecnologia è ancora in fase di sperimentazione e quello che oggi si trova in commercio sono ancora prodotti sperimentali. Si stima che i lettori base costeranno all'incirca $ 80,00 e che i tag arriveranno a costare pochi centesimi. Sulla base di queste considerazioni, è ovvio che deve essere fatta una accurata stima prima di decidere di implementare questa tecnologia. Si rischia infatti che, per risparmiare sui costi di inventario, si spendano inutilmente parecchie migliaia di euro. Il settore in cui la tecnologia RFID diventa quasi indispensabile è quello della distribuzione. Si provi ad immagine un grosso centro di distribuzione che quotidianamente movimenta decina di migliaia di prodotti.

I costi di implementazione di tutta l'infrastruttura sono ammortizzabili in tempi proporzionalmente brevi ed i vantaggi derivanti sarebbero superiori al denaro investito. È invece sconsigliato l'utilizzo di questa tecnologia ove i volumi di merce movimentata sono bassi. Ad esempio, se produciamo magliette e le vendiamo ai negozi della nostra città, è inutile investire nella tecnologia RFID. Gli stessi vantaggi li otterremo con i semplici tag visti nei precedenti articoli.

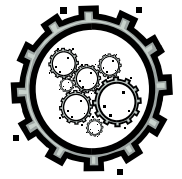
CONCLUSIONI

Si conclude qui la serie di tre articoli riguardanti la tecnologia dei tag a radiofrequenza.

Dalla tecnologia RFID, sebbene sia pensata per le grosse compagnie (basta vedere chi fa parte del consorzio EPCglobal per farsi un'idea), ne trarranno vantaggi un po tutti.

Da parte nostra, come sviluppatori, non possiamo che prepararci a far fronte alle richieste, che di sicuro non mancheranno, dei nostri clienti.

Buon lavoro.

Michele Locuratolo

SUL WEB

www.sun.com/software/solutions/rfid/index.html

www.rfidexchange.com/forum/default.asp

www.verisign.com/nds/directory/epc/fyi/howitworks.html

www.aimglobal.org/standards/rfidstds/sc31.asp

www.rfidjournal.com

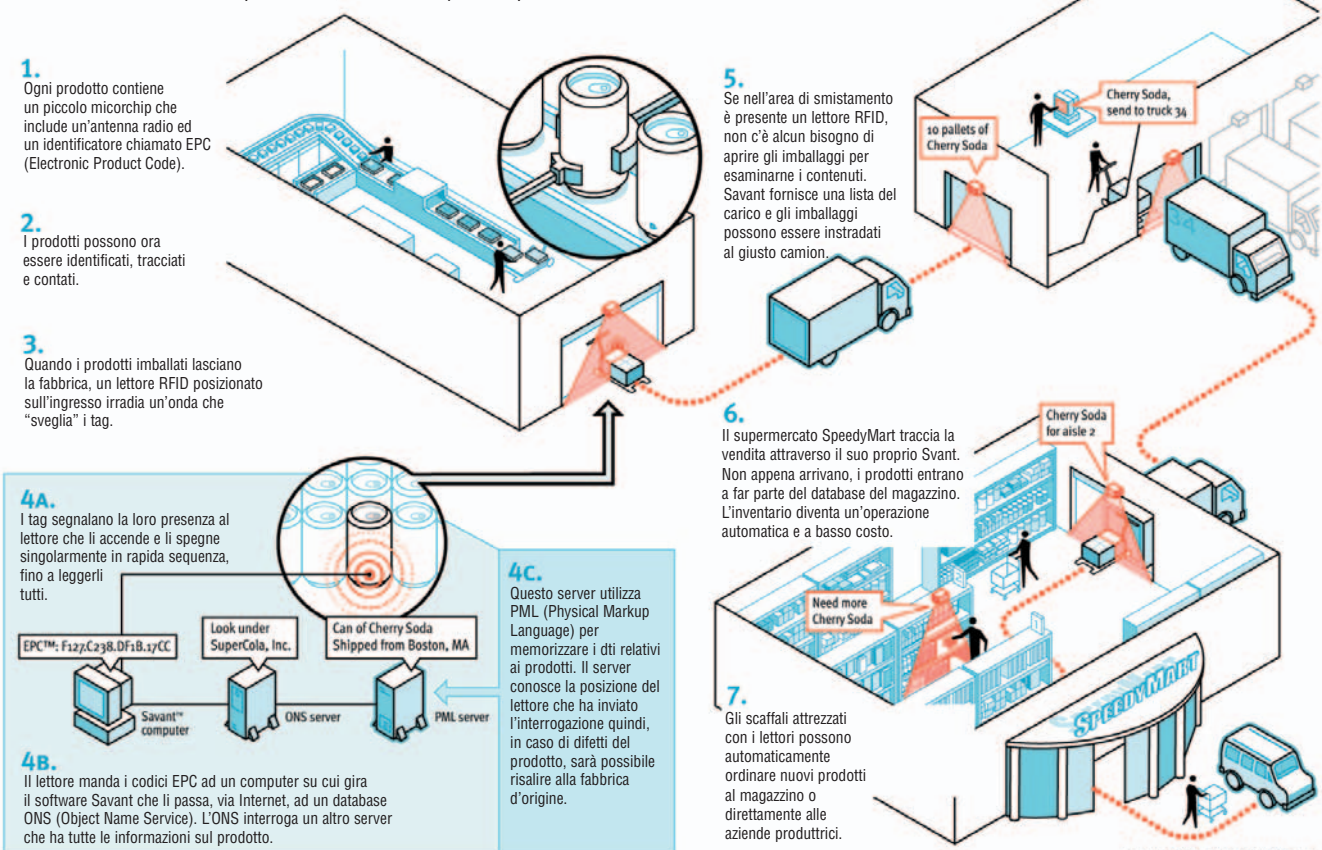



Fig. 4: ***Rappresentazione grafica del ciclo di vita dei prodotti***

VBA e Access per costruire applicazioni

Test di certificazione realizzati in Access ultima parte

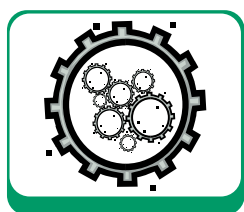
Dopo aver analizzato a cosa serve la maschera di configurazione in questo nuovo appuntamento concluderemo il discorso mostrando le parti di codice che compongono l'applicazione

Nel momento in cui *Maschera configurazione* richiama *Maschera Test*, abbiamo visto che le passa diversi parametri. Attraverso essi vengono impostate delle variabili opportune e, soprattutto, viene costruito il filtro che consentirà di selezionare le sole domande facenti parte del set di argomenti prescelti. In particolare, abbiamo visto che accadono altre due cose importanti. La prima è che viene richiamata una procedura, *Random()*, alla quale vengono passati dei parametri che le consentiranno di riempire "correttamente" l'array di strutture *Listarecords*. Questa struttura, costituita da un numero di elementi pari al numero di domande generate, abbiamo visto essere in realtà un array di strutture *Record* che, tra le altre informazioni, contengono il numero di record "effettivo" all'interno di RS. Il secondo particolare importante del codice mostrato la volta scorsa è l'inizializzazione di altre due variabili, *Cursore* e *LastCursore*, operazione questa che precede lo spostamento all'interno del recordset *RS*. Per poter capire bene a cosa servano queste due variabili e soprattutto perché sono state introdotte, è bene fare alcune considerazioni importanti. Durante il corso di un test, il candidato che si trova davanti una domanda qualsiasi, può compiere almeno tre azioni:

Fig. 1: La maschera "Maschera Test"

- impostare il bookmark sulla domanda;
- rispondere alla domanda;
- rispondere alla domanda ed impostare il bookmark.

Soltanto al verificarsi di una di queste condizioni, vengono abilitati i pulsanti *Successivo* ed, eventualmente, il pulsante *Precedente* (se esistono bookmark precedenti alla domanda corrente). A questo punto, il candidato può decidere se spostarsi sulla successiva domanda o passare a rivedere i bookmark precedenti. In quest'ultimo caso è ovvio che, se l'utente si sposta avanti ed indietro tra un bookmark e l'altro, ad un certo punto, quando desidererà passare alla successiva domanda del test (ossia dal punto in cui aveva "abbandonato" la prosecuzione dell'esame), non si saprebbe più su quale record riposizionarsi. Peraltro, è anche bene sottolineare che l'ultimo bookmark impostato dall'utente non deve necessariamente coincidere con l'ultima domanda *N* alla quale ha risposto, pertanto la pressione del pulsante *Successivo* deve provocare uno spostamento non sul record *X+1*, ma sull'*N+1*.
Tutto ciò spiega il perché si sia resa necessaria l'introduzione di queste due variabili che ci consentono di gestire correttamente gli spostamenti tra un bookmark e l'altro, senza perdere l'informazione relativa alla posizione sulla quale eravamo prima di "muoverci". La variabile *Cursore* conserva sempre, ad ogni passaggio da un record di *RS* all'altro, l'indice che indica la posizione "attuale" del record sul quale ci si trova in quel momento. La seconda, invece, conserva la posizione sulla quale ci si trovava un istante prima dello spostamento. Queste due variabili, in realtà, non memorizzano numeri di record, ma valori che costituiscono un indice all'interno di *ListaRecords* attraverso cui, mediante l'item *num_rec* corrispondente, possiamo accedere al record di *RS* vero e proprio. Quindi, quando diciamo che *Cursore* ha valore quattro, ad esempio, stiamo sottintendendo che "ci troviamo sul record indicato dall'item *num_rec* della terza struttura *Record* di *ListaRecords()*". La corretta gestione dei bookmark è stata senza dubbio una delle parti più complicate da gestire. Infatti, i vincoli al progetto che ci eravamo imposti, prevedevano che l'utente potesse spostarsi sui bookmark definiti in precedenza in qualunque momento del test e pertanto occorreva trovare un si-

stema che permettesse di compiere queste azioni senza perdere nessuna di queste informazioni. In definitiva, dunque, il codice che gestisce tutto ciò, è racchiuso all'interno di due procedure che gestiscono l'evento *Click()* dei due pulsanti *Precedente* e *Successivo* della maschera *Maschera Test*. Qui analizzeremo soltanto il codice "allegato" al pulsante *Successivo* poiché risulta leggermente più complesso dell'altro (dovendo gestire non solo lo spostamento sui bookmark, ma anche quello sulla successiva domanda, oltre alla gestione dei risultati) e soprattutto perché la logica che sta dietro è comunque la stessa. Di seguito ecco un estratto dal progetto:

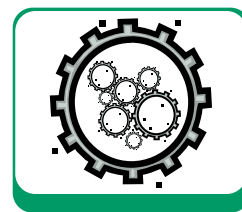
```
Private Sub Successivo_Click()
Dim RS As DAO.Recordset ' Recordset delle domande
Dim i, j As Integre             ' Contatori
Dim BookTrovatoPrec As Boolean ' Serve per informare
                 se è stato trovato un bookmark successivo
Dim BookTrovatoSucc As Boolean  ' Serve per informare
                  se è stato trovato un bookmark precedente
Dim NumDomande As Integer ' Numero di domande
                                                del test
...
          End If
 End If
RS.AbsolutePosition=ListaRecords(Cursore).num_rec
ReimpostaRisposte
End Sub
```

La comprensione di queste righe di codice, dopo le premesse fatte ed i commenti inseriti, crediamo sia abbastanza semplice. Ogni qualvolta l'utente preme il pulsante *Successivo*, vengono effettuati innanzitutto due controlli. Il primo consente di vedere se esistono domande precedenti per le quali l'utente ha impostato il bookmark, in maniera tale da riabilitare il pulsante *Precedente*. Il secondo controllo, invece, permette di vedere se esistono bookmark successivi a quello corrente. Se questa condizione è vera, significa che bisogna passare sulla successiva domanda che soddisfa questa condizione (ossia sul successivo bookmark), altrimenti possiamo passare al punto in cui eravamo, ossia alla nuova domanda del test. Superati questi due controlli, la procedura verifica se siamo arrivati al termine del test, ossia se *Cursore* è uguale al numero di domande. Se quest'ipotesi è soddisfatta, viene innescato un ulteriore controllo che consente di stabilire se tutte le domande dell'esame sono prive di bookmark e, soprattutto, se l'utente ha risposto realmente a tutto il test. Quest'ulteriore verifica è necessaria poiché abbiamo stabilito che l'utente, quando si trova davanti una nuova domanda, affinché possa "riabilitare" il pulsante *Successivo* e, quindi, passare alla domanda seguente, deve aver dato almeno una risposta o aver impostato il bookmark sulla domanda. Ovviamente, probabilmente è inutile dirlo, l'utente può sia rispondere sia impostare il bookmark su di una domanda allo stesso tempo. La verifica di questa condizione avviene attraverso un'operazione logica che coinvolge anche le due variabili booleane *BookTrovatoPrec* e *BookTrovatoSucc,* valorizzate nei due controlli visti precedentemente. Se non esistono bookmark, allora il programma "dichiara" terminato il test e mostra i risultati. Se, al contrario, il programma rileva che esistono ancora bookmark, avverte opportunamente l'utente con un messaggio a video, invitandolo a rivedere le domande. Solo al termine di queste operazioni, ossia quando il candidato ha risposto a tutte le domande ed ha rimosso tutti i bookmark, si procede con il calcolo del risultato.

LA GESTIONE DELLE DOMANDE

Il secondo aspetto complicato dell'intero progetto è stato senza dubbio quello della corretta "simulazione" dei diversi tipi di domande possibili. Come accennato la volta scorsa, possono esistere due tipi di domande possibili:

- a risposta singola;
- a risposta multipla.

Le domande a risposta singola prevedono la possibilità di scegliere solo una risposta all'interno delle cinque disponibili, mentre le altre concedono all'utente più di una scelta. Tuttavia, quest'ultima categoria può essere suddivisa in ulteriori due tipi: quelle che prevedono un numero (maggiore di uno) di risposte prefissato e quelle che lasciano che sia l'utente a decidere quante, tra quelle disponibili, sono corrette. La distinzione tra una domanda e l'altra ed il numero di risposte possibili è riconducibile a due campi del database: *RispostaMultipla* e *NumMaxRisposte*. Il primo non è altro che un flag che stabilisce se le risposte possono essere solo una o più di una. Il secondo, invece, memorizza il numero di risposte possibili. Per convenzione, si è assunto che, se il valore di quest'ultimo campo è pari a cinque, la domanda prevede un numero di risposte multiple imprecisato, ossia può essere stabilito dall'utente. Per essere maggiormente chiari, in questa categoria finiscono tutte quelle domande che chiedono "…selezionare tutte le risposte possibili". A questo punto, stabilita questa distinzione, appare ovvio che la via più naturale per mostrare a video le due diverse tipologie di domande, è quella di sfruttare dei controlli di tipo *Option Button* per quelle a risposta singola e dei controlli di tipo *CheckBox* nei restanti casi. Per non dover essere costretti a gestire due insiemi di controlli diversi e, quindi, il doppio di eventi possibili, si è scelta una via "alternativa" che mostra, per tutti i tipi di domande, le possibili risposte

MODALITÀ ADAPTIVE

Questa modalità, come già anticipato nella prima parte, consente la generazione di ogni domanda a "runtime" ossia, ad ogni risposta dell'utente, "intuisce" quali siano le lacune del candidato e si comporta di conseguenza. Ovviamente, l'implementazione di questa modalità necessita di una riscrittura parziale del codice, soprattutto nella parte di estrazione casuale delle domande da DBTest e popolazione del recordset RS.

GESTIONE DELLE DOMANDE A RISPOSTA MULTIPLA

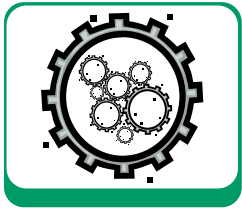
Malgrado il meccanismo di gestione delle risposte dell'utente, attraverso i soli controlli *Option Button*, possa risultare efficiente, qualcuno potrebbe non "gradire" questa scelta. Potete modificare questo comportamento gestendo "separatamente" i diversi tipi di domanda, ossia inserendo, laddove previsto, i controlli di tipo *CheckBox*.

sotto forma di *Option Button* e gestisce la loro selezione in maniera differente a seconda della domanda stessa. Vediamo come funziona questo meccanismo. Sulla maschera *Maschera Test* sono posizionati, proprio sotto al controllo che mostra il testo della domanda corrente, cinque *Option Button* denominati *OpzioneA, OpzioneB, OpzioneC, OpzioneD* ed *OpzioneE*. Ogni qualvolta l'utente seleziona (fa click con il mouse) uno di questi controlli per "accendere" quella determinata risposta, viene invocata la procedura *ControllaRisposte* e le viene passata la lettera corrispondente all'identificativo della risposta (ad esempio, se si preme su *OpzioneA*, viene passata alla procedura la lettera *A* e così via). Questa subroutine è così strutturata:

Fig. 2: **La maschera che mostra l'esito del test**

```
Private Sub ControllaRisposte(RispostaUtente As String)
Dim RispostaData As String
Dim max As Byte
Dim Count As Byte
...
 ListaRecords(Cursore).Risposta=RispostaData
 ListaRecords(Cursore).Corretta=(RispostaData=
                        Me.Recordset.Fields!RispostaEsatta)
End Sub
```

La prima parte della procedura verifica il tipo di domanda. Se la risposta deve essere unica (il campo *RispostaMultipla* del record corrente è *False*), allora abilita (imposta a *True*) il valore dell'*Option Button* selezionato. Se, invece, *RispostaMultipla* è *True*, allora inserisce, all'interno della variabile *RispostaData* ed in ordine alfabetico, le lettere corrispondenti agli *Option Button* sino a quel momento selezionati. Infine, controlla che l'utente abbia selezionato un numero adeguato di risposte ed abilita, di conseguenza, il pulsante *Successivo* per consentire il passaggio alla nuova domanda. A questo punto, memorizza nell'item *Risposta* della struttura *Record* "corrispondente" la risposta dell'utente e valorizza adeguatamente l'item *Corretta*. Quando abbiamo analizzato il codice relativo all'evento *Click()* del pulsante *Successivo*, qualcuno probabilmente si sarà accorto della presenza di un'istruzione

Fig. 3: **Le maschere "Maschera Update Exams" e "Esami"**

che richiama una procedura denominata *ReimpostaRisposte*. Essa non fa altro che leggere il valore dell'item *Risposta*, scorporarlo e valorizzare di conseguenza le *Option Button* "interessate". La procedura *ReimpostaRisposte* è molto semplice:

```
Sub ReimpostaRisposte()
Dim i As Integer
Dim PorzioneRisposta As String
Me.OpzioneA=False
Me.OpzioneB=False
...
Me.OpzioneE=False
For i=1 To Trim(Len(ListaRecords(Cursore).Risposta)) Step 2
  PorzioneRisposta=Mid(ListaRecords(Cursore).Risposta, i, 1)
  Select Case PorzioneRisposta
       Case "A":   Me.OpzioneA.Value=True
       Case "B":   Me.OpzioneB.Value=True
       Case "C":   Me.OpzioneC.Value=True
       Case "D":   Me.OpzioneD.Value=True
       Case "E":   Me.OpzioneE.Value=True
  End Select
Next
```

Dopo aver "azzerato" tutte le risposte, essa si serve di *ListaRecords* per recuperare quelle date dall'utente, impostando a *True* i corrispondenti *Option Button*.

ULTERIORI ASPETTI DI CERT

Il programma è in grado di gestire test a tempo ossia simulazioni che hanno una durata stabilita dal candidato attraverso la maschera di configurazione. La gestione di test di questo genere avviene "semplicemente" attraverso l'implementazione del codice necessario inserito all'interno dell'evento *Timer()* di *Maschera Test:*

```
Private Sub Form_Timer()
Dim TempoRimanente As Integer
Dim NumDomande As Integer
Dim i As Integer
Dim stDocName As String
If Tempo>0 Then
   NumDomande=UBound(ListaRecords)
   TempoRimanente=Tempo-Int((Timer-OraAttuale)/60)
   LblTimer.Caption="Il tempo a tua dispozione è: "
                          & TempoRimanente & " minuti"
   If TempoRimanente=0 Then
      MsgBox "E' terminato il tempo a tua disposizione!!!"
      MostraRisultati
   End If
End If
End Sub
```

Al termine dei minuti stabiliti, il test si blocca e mo-

stra a video un'opportuno messaggio che avverte l'utente che il tempo a sua disposizione è terminato. Successivamente, il programma mostra il risultato del test, attraverso il richiamo di una procedura denominata *MostraRisultati* che, tra le altre cose, viene sfruttata anche all'interno dell'evento *Click()* del pulsante Successivo. I risultati del test sono mostrati a video evidenziando non solo la percentuale appena raggiunta, confrontata con il *Passing Score* impostato, ma anche l'elenco delle domande alle quali l'utente ha risposto in maniera corretta e quelle che ha sbagliato.

```
Sub MostraRisultati()
  Dim RisposteEsatte As Integer
  Dim NumDomande As Integer
  RisposteEsatte=0
  NumDomande=UBound(ListaRecords)
  ' Calcolo delle risposte esatte...
   For i=0 To NumDomande-1
       If ListaRecords(i).Corretta Then
                    RisposteEsatte=RisposteEsatte+1
   Next
  ' Calcolo percentuale: RisposteEsatte:
                 NumDomande=Percentuale:100
   Percentuale=Round((RisposteEsatte *
                        100)/NumDomande, 0)
  ' Prelevo l'ultimo elemento dell'array contenente il
                                passing Score...
   PassingScore=Val(ArrayOfArgs(ArrayLength-1))
   DoCmd.Close
  ' Mostra il risultato del test
   stDocName="Risultato Test"
   DoCmd.OpenForm stDocName,,,,,, Percentuale &
                              ";" & PassingScore
End Sub
```

Terminata la parte relativa alla simulazione di un test di certificazione, restano da vedere soltanto altre due cose: la modifica o l'inserimento di nuovi esami e nuove domande e la generazione dei test su supporto cartaceo. Per il primo problema sono state predisposte tre maschere, chiamate banalmente *Maschera Update Exams*, *Esami* e *Domande*. La prima rappresenta un semplice pannello di controllo attraverso il quale è possibile richiamare le restanti due. Attraverso il primo pulsante, viene richiamata la maschera *Esami* che consente all'utente di aggiornare il database con nuovi codici di esame e nuovi argomenti. Il secondo pulsante, invece, permette l'inserimento o la modifica di nuove domande. Per quanto riguarda quest'ultimo aspetto è bene sottolineare che le modalità d'immissione di nuove domande sono "assolutamente" manuali, nel senso che l'utente deve inserire, una per una, ogni domanda e le relative risposte, preoccupandosi di valorizzare soprattutto i campi del DB che conservano il tipo di domanda, l'ID dell'esame, la risposta esatta, ecc. Ovviamente possono esistere sistemi più pratici e comodi per raggiungere questo obiettivo, ma al momento non abbiamo previsto alcuna procedura automatizzata che consenta di rendere più agevole questo compito. Tuttavia, è bene ricordare che Microsoft Access offre diversi modi per importare dati all'interno di una tabella e che, uno di questi, ad esempio, prevede l'inserimento automatico, gestibile anche da codice, di un file *.CSV*. Detto questo, ci resta solo un ultimo aspetto da considerare e, quindi, passiamo ad illustrare brevemente come avviene la generazione dei test su supporto cartaceo. Quando l'utente preme il secondo pulsante della maschera di configurazione, viene innescata una serie di azioni molto simili a quanto visto già per la generazione dei test a video, con la differenza che i dati (record) selezionati sono aggiunti all'interno di una apposita tabella denominata *TempDBTest*. Questa base dati funge da appoggio per un report, *ReportTest* che consente di gestire domande, risposte e risultati su carta. Una volta che la tabella suddetta è riempita con i record del test, viene mostrata a video un'anteprima del report e l'utente può decidere se stamparla oppure no.

Fig. 4: La maschera Domande

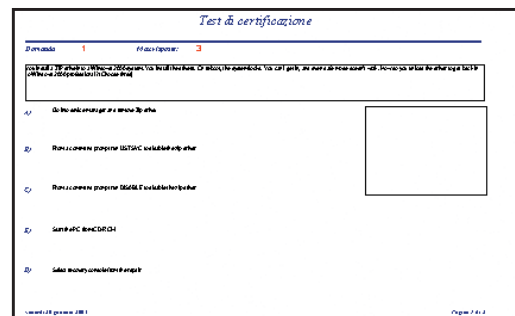



Fig. 5: Il report ReportTest

```
Private Sub CmdGeneraReport_Click()
On Error GoTo Err_CmdErrore_Click
Dim NumeroDomande As Integer
Dim StringaArgomenti As String
Dim stDocName As String
Dim stLinkCriteria As String
Dim i As Integer
Dim varItm As Variant
...
RSTemp.Update
Next
stDocName="ReportTest"
DoCmd.OpenReport stDocName, acViewPreview
Exit_CmdErrore_Click:
    Exit Sub
Err_CmdErrore_Click:
    MsgBox Err.Description
    Resume Exit_CmdErrore_Click
End Sub
```

Alberto Lippo e Francesco Lippo

NOTA

HISTORY

Potrebbe essere interessante prevedere di conservare, magari in un file di testo, i miglioramenti sinora fatti (rispetto ai test precedenti) di un determinato candidato. Ovviamente, il programma dovrebbe prevedere, all'avvio, l'avvio di una procedura per la richiesta del nome dell'utente e memorizzare, al termine della simulazione, le informazioni storiche più importanti, come la data del test, il risultato raggiunto, passing score, durata dell'esame, numero di domande, ID dell'esame, ecc.

Le regole per partecipare alla nuova edizione di Crobot

Crobots XIV anno

Di come ogni anno venga scelto il fortunato estensore degli articoli sul nostro gioco preferito si è già parlato la scorsa volta È arrivato il momento di introdurre le regole della sfida

La settimana scorsa, quando il buon Maurizio Camangi mi ha fatto presente che era il tempo di indire il concorso 2004, a malincuore gli ho risposto:

S: Bene Joshua, chi scegliamo come articolista quest'anno?
J: Ma te, naturalmente.
S: Cavoli Maurizio, non si può fare. Oramai io ho vinto un torneo.
J: E secondo te dove lo troviamo un altro così fuori di testa?

Non ho certo potuto contraddirlo. E pertanto eccomi qui, ad annoiarvi ancora una volta con mostriciattoli che si divertono a spararsi in un'arena virtuale. Dovrei, naturalmente, fare il punto della situazione, spiegare cosa è successo da dicembre ad oggi, analizzare il comportamento dei combattenti che hanno dato vita ai due tornei ufficiosi che hanno incoronato il robot più forte di sempre (sia secondo la modalità classica con eliminatorie e finale a 32, sia secondo una formula sperimentale con un'ultima fase da 96 partecipanti). Tuttavia, quando si tratta di parlare di robot, soprattutto se si tratta dei vincitori, io ho una tecnica particolare. Infatti, non ne parlo affatto. È inutile che dica quanto sono stati bravi i programmatori: se sono arrivati in finale, devono per forza avere svolto un ottimo lavoro. Inoltre non è mai una cosa simpatica per il mio amor proprio commentare codice che, per forza di cose, è enormemente migliore e più elegante di quello che potrò mai scrivere io. Quindi mi intrattengo a raccontare vita, morte e miracoli degli autori. Dato che, bene o male, ci conosciamo tutti, argomenti da trattare ce ne sono sempre in abbondanza e, in teoria, le probabilità di non trovare nulla da dire sono infinitesimali. Ma le teorie sono fatte apposta per essere confutate e se un bel diavoletto ci mette lo zampino, può capitare che non uno, ma ben tre programmatori mi siano assolutamente ignoti. E il caso ha voluto che siano gli artefici dei contributi più interessanti di quest'ultimo anno: Michele Cardinale, Roberto e Martino Candon. Il primo ha elaborato un'analisi dettagliatissima del funzionamento delle funzioni di fuoco di Tox.r (che, per inciso, sembrano essere tornate di moda nella scorsa edizione del torneo. Dateci uno sguardo). Gli altri due, padre e figlio dodicenne al seguito, sono stati la vera sorpresa del 2003, piazzando numerosi robot in tutte le finali. Non contenti, hanno poi animato la Mailing list sviscerando, come mai era avvenuto prima, il comportamento e le tattiche di buona parte dei partecipanti alla manifestazione. Una notevole mole di lavoro, che dovrebbe dare i suoi frutti nel Torneo di Crobots 2k4 oramai alle porte! La data ufficiale è ancora da stabilire con sicurezza (controllate gli eventuali aggiornamenti sul sito *www.ioprogrammo.it/crobots*) ma è quasi certo che, tempi tecnici permettendo, le iscrizioni si chiuderanno nuovamente alla mezzanotte del 31 ottobre 2004. Per il secondo anno consecutivo le regole non mutano,

NOTA

I PREMI IN PALIO

1° PREMIO
- **WL-122 Wireless Network Broadband Router 100g+ (100 Mbps)**
- **WL-107 Wireless Network USB Adapter 100g+ (100Mbps)**
- **WL-121 Wireless Network PCI card 100G+ (100 Mbps)**

2° PREMIO
- **CN-651 USB 2.0 Memory Drive 256 MB**

3° PREMIO
- **CN-126 USB 5.1 Audio Adapter**

visto il loro impeccabile funzionamento nelle passate edizioni. Gli unici dettagli ancora da stabilire sono, pertanto:

- Il numero di combattenti che si sfideranno nella finale;
- Il peso da assegnare allo scontro uno contro uno rispetto a quello 4vs4. È allo studio la possibilità di farlo passare dal rapporto di 1:5 a quello di 1:4. Fateci sapere che ne pensate in Mailing List.

Il bando ufficiale sarà consultabile, naturalmente, sui vari siti, ufficiali o meno, dedicati alla manifestazione. Per comodità riporto di seguito un breve elenco dei punti salienti.

MODALITÀ DI ISCRIZIONE

È possibile inviare fino a 4 crobots alla competizione:

1. Uno dovrà rientrare al di sotto della soglia delle 500 istruzioni comprese e parteciperà al torneo dei MicroRobot.
2. Uno avrà a disposizione 1000 Vb di codice, e potrà essere iscritto alla categoria ClassicBots (in cui confluiranno anche i partecipanti alla sezione precedente).
3. Il terzo potrà sfruttare l'intero limite di indirizzamento consentito dal compilatore (2000 VirtuaByte) e disputerà il Torneo Generale (nel quale si sfideranno i robot di tutte e tre le classi di peso).
4. Il quarto concorrente, infine, sarà a discrezione del programmatore, che potrà scegliere in quale delle tre precedenti categorie far rientrare la sua opera. Il numero dei robot non è, ovviamente, vincolante: fermo restando il limite superiore di quattro unità per autore, è possibile partecipare all'evento anche con un unico combattente.

MODALITÀ DI COMBATTIMENTO (PROVVISORIE)

- I robot andranno spediti all'indirizzo di posta elettronica *ascheri@edmaster.it* (oppure inviati con gli altri mezzi previsti nel bando).
- Tutti i robot arrivati saranno divisi in gruppi di al massimo 32 programmi (sempre nel caso, naturalmente, che il loro numero ecceda tale cifra), che disputeranno i gironi eliminatori. Questi forniranno i nomi, sia direttamente che tramite ripescaggio, dei qualificati per la fase finale.

I crobot si affronteranno, in tutte le fasi della competizione, secondo due modalità differenti:

1. Nello scontro 4vs4, naturalmente, ad ogni match partecipano quattro robot. Ogni sfidante deve incontrare ciascuno degli avversari almeno 2000 volte: il fattore di ripetizione varierà quindi a seconda dell'ampiezza del girone.
2. Nello scontro F2F, invece, ogni robot affronta a singolar tenzone ciascuno dei componenti del girone, con un fattore di ripetizione pari a 2000.

Per ridurre la durata degli incontri, viene posto il "limite temporale" dei 200.000 cicli virtuali di CPU, al termine dei quali la partita viene dichiarata patta tra i sopravvissuti.
I punteggi saranno assegnati secondo lo schema *Pranzo*:

- 12 punti al robot vittorioso;
- 3 punti ai superstiti di un pareggio a due;
- 2 punti ai superstiti di un pareggio a tre;
- 1 punto ai superstiti di un pareggio a quattro con danni superiori al 40%;
- 0 punti ai superstiti di un pareggio a quattro con danni inferiori al 40%;
- La classifica finale uscirà dal mixaggio tra le due graduatorie, con pesi 5*4vs4 e 1*F2F, per sottolineare la difficoltà dello scontro a 4.
- La finale sarà disputata secondo le stesse regole.

Il dado è tratto, la sfida è indetta. Sono certo che l'ambita palma della vittoria basterebbe a ripagare i croboticisti delle loro fatiche ma, dato che un riconoscimento più materiale sgradito non lo è mai, anche questa volta ioProgrammo, patrocinatore dell'evento, mette in palio dei premi per il vincitore della competizione.
Sono sicuro che, accanto a quanti hanno animato la mailing listi in questi ultimi mesi, anche questa edizione del torneo vedrà la partecipazione di numerosi veterani. Alessandro Carlin, Daniele Nuzzo, Andrea Creola (fresco microcampione, oltretutto) non si faranno scappare la possibilità di riagguantare uno dei titoli in palio.
Sarà interessante verificare se riusciremo, una volta di più, a mantenere il trend positivo relativamente alle matricole. Chi volesse prendere spunto dal lavoro svolto negli anni precedenti, potrà ispirarsi all'enorme quantità di robot inviati negli ultimi 14 anni, ripartiti nelle varie classi in cui è organizzata la manifestazione: 500, 1000 e 2000 istruzioni. Visitando i siti dedicati a Crobots per trovare tutto il materiale necessario allo sviluppo di un combattente, non dimenticate di scaricare anche alcuni programmi accessori che velocizzano enormemente la fase di sviluppo: Torneo XP, Count 8.3 (per calcolare le classifiche e effettuare analisi statistiche sui risultati), *Ecat* (che ricerca automaticamente i parametri migliori per il vostro robot), *PPC* (preprocessore di comandi che aggiunge *#include* e *#define* al compilatore) *CRH2.0* e il suo antagonista, *TorneoGUI* (comode interfacce visuali per i comandi da console visti prima).
Direi che anche per questa volta abbiamo terminato. Con la speranza di superare il record di partecipazioni stabilito lo scorso anno faccio a tutti quanti un grosso in bocca al lupo: iscrivetevi alla mailing list ufficiale di crobots (*crobots@ioprogrammo.it*) e… fatevi sotto!!!!

Simone Ascheri
Maurizio Camangi
Michelangelo Messina

Un'introduzione al nuovo IDE open-source

Sviluppare in Java con Eclipse 3

parte prima

È da poco uscita l'ultima versione di Eclipse ed ioProgrammo ha deciso di renderle omaggio con questa miniserie che vi guiderà nell'uso delle sue innumerevoli caratteristiche e funzionalità

Con l'uscita della sua terza versione, Eclipse ha fatto veramente passi da gigante. È sempre stato un ottimo tool per lo sviluppo in Java, affidabile e professionale, ma ora si è aggiudicato probabilmente il primo posto nella rosa degli IDE per il linguaggio più diffuso del momento. Tra le nuove caratteristiche che troviamo nell'ultima release abbiamo un *miglioratolook* dell'applicazione (con eleganti tab dal bordo rotondo), la possibilità di eseguire in background alcuni task (approccio detto *Responsive UI*), l'esistenza di wizard che oltre a guidarti in alcune procedure chiave possono anche compiere delle operazioni esemplificative (i cosiddetti *Cheat Sheet*), *Find/Replace* con espressioni regolari, e molte altre caratteristiche che tutti gli utenti, vecchi e nuovi, apprezzeranno per l'aiuto che danno a chi sviluppa e gestisce progetti di programmazione Java. Molti dei miglioramenti che sono stati apportati non sono direttamente visibili all'utente e riguardano la struttura interna del prodotto. Da un lato, l'API per la creazione dei plug-in è stata riscritta interamente con la conseguenza di migliorare il supporto per le estensioni del framework e fornire più flessibilità ed opzioni a chi le sviluppa. Da un altro lato, e come parte di tale reengineering, lo stesso sostrato infrastrutturale di Eclipse è stato rivisto alla luce di quello che ora è chiamato *Rich Client Platform* (RCP): sebbene *Eclipse* nasca come base per la creazione di ambienti integrati di sviluppo (IDE), grazie alla totale revisione della sua architettura oggi il workbench può essere utilizzato come supporto per lo sviluppo di qualunque tipo di applicazione, sfruttando però tutte quelle caratteristiche che sono a disposizione di chi scrive plug-in.
Insomma, si potranno creare applicativi gestionali, grafici, di interazione, e quant'altro, basandosi sull'approccio nativo di Eclipse di prospettive, *view* ed *editor*. Per chi si interessa ad Eclipse per le sue potenzialità come editor ed IDE di sviluppo Java, dedichiamo le pagine di questa breve serie ad una dettagliata descrizione delle caratteristiche dell'ambiente di sviluppo, nel tentativo di farvi apprezzare quanto sia semplice, intuitivo e comodo lavorare con questo capolavoro frutto degli sforzi della comunità open-source. Cominceremo dall'inizio, permettendo anche a chi non ha mai lavorato con Eclipse 2.1 o precedenti di seguire agevolmente il filo del discorso: chi è già un fan appassionato, invece, scoprirà le novità introdotte nella versione 3.0 insieme ad un ripasso di quanto c'era da prima.

WORKBENCH E WORKSPACE

Il workbench è la finestra principale di Eclipse dove svolgiamo il nostro lavoro di editazione del codice, gestione delle risorse applicative e tutto quanto il prodotto ci permette di fare. Il workspace invece è la directory sotto la quale è raccolto il nostro materiale, intendendo con questo sia i file Java, che le classi compilate, che tutti gli eventuali file utilizzati per le nostre applicazioni. Inoltre all'interno del workspace troviamo anche la cartella *.metadata*, con i file di configurazione ed i file di log del workbench e dei

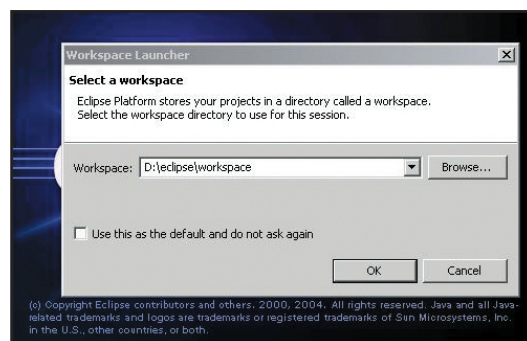



Fig. 1: **La dialog box di scelta della cartella per il workspace**

vari plug-in caricati insieme ad esso e all'interno delle sottocartelle dei nostri progetti, alcuni file di informazione supplementari utilizzati da Eclipse internamente per la gestione dello spazio di lavoro. All'avvio di Eclipse viene per prima cosa richiesto quale sarà la cartella del workspace (come in Fig. 1). Non è necessario che la cartella esista se si sta definendo un nuovo workspace, ci penserà automaticamente Eclipse a crearla. Scelto il punto di raccolta del vostro materiale, si aprirà la finestra del workbench e potrete cominciare a lavorare. È possibile aprire diverse istanze di tale finestra (per esempio se lavorate con due monitor) così come è possibile eseguire più volte Eclipse per avere istanze che fanno riferimento a workspace differenti (utilissimo per chi sviluppa plug-in per il framework). La prima finestra ad accogliervi sarà giustamente quella di benvenuto (*"Welcome"*, visibile in Fig. 2) dalla quale potete ottenere qualche aiuto per iniziare a muovervi nell'ambiente o passare direttamente al sodo con il link *"Go to the workbench"*.

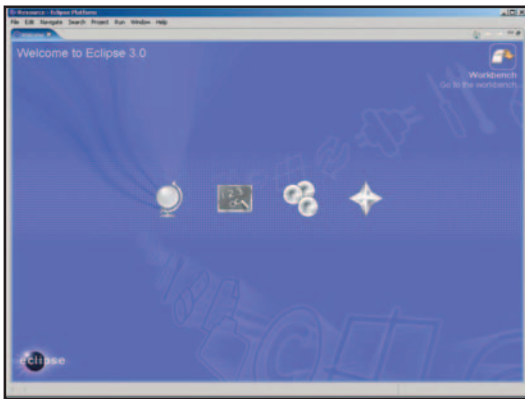



Fig. 2: **La finestra di benvenuto mostrata all'avvio di Eclipse**

PERSPECTIVE, VIEW ED EDITOR

La finestra di lavoro rappresentata dal workbench è un po' come un contenitore: offre solo un comodo ambiente in cui i vari plug-in possono svolgere i loro compiti per venire incontro alle esigenze dello sviluppatore. Il contenuto del workbench sono le perspective (una o anche più in contemporanea), le quali a loro volta (un po' come le scatole cinesi) sono anche dei contenitori pronti solo ad accogliere elementi visivi. L'idea della prospettiva è quella di mettere a disposizione dell'utente una serie di strumenti per portare a termine determinate funzionalità, aggregando nello spazio offerto dal workbench una serie di sottofinestre (genericamente dette *part*) che si concentrano su una tipologia di attività specifica. Le *part* sono suddivise in due tipi, a seconda del loro obiettivo: possono servire per scorrere e navigare tra le risorse (le *view*), oppure per visualizzarle in dettaglio e/o modificarle (gli editor). La prospettiva di default al primo avvio è *"Resource"*, la quale contiene una view per navigare il filesystem del workspace (*"Navigator"*, concettualmente simile all'Explorer di Windows), una per la gestione dei nostri task (*"Task"*) ed una vista che mostra la struttura di eventuali file aperti nell'editor (*"Outline"*, che – per esempio – dà una visualizzazione gerarchica dei membri di una classe). L'area centrale di una prospettiva è tipicamente dedicata agli editor, i quali vengono invocati selettivamente facendo doppio-click su un elemento visualizzato in una view. La loro posizione resta ancorata allo spazio loro dedicato, mentre per le view l'utente può scegliere tranquillamente come spostare e muovere le varie finestre in base ai propri gusti. Le view possono anche essere utilizzate in modalità *Fast View*, simile ai toolbox unpinned di Visual Studio .NET che si autonascondono quando si clicca al di fuori della loro finestra. Per sperimentarla, selezionate l'omonima voce dal menu contestuale che appare cliccando col destro sulla barra del titolo di una view qualunque. Per quanto riguarda le prospettive, è possibile personalizzarle scegliendo quali viste esse contengono e in quale posizione. Volendo si può anche salvare un proprio layout, creando così una nuova prospettiva. Sentitevi pure liberi di sperimentare con le varie possibilità che il workbench offre: se fate dei pasticci senza rimedio, potete sempre tornare ai settaggi originali delle prospettive integrate con il menu *Window->Reset Perspective*. Il workbench, poi, vi permette di tenere aperte più prospettive contemporaneamente richiamandole dal menu *Window-> Open Perspective:* una struttura simile alla taskbar di Windows in alto a destra vi permetterà di passare tra una e l'altra, oltre ad un pulsante iniziale (quello con il "+") che replica la funzionalità del menu appena menzionato. In questa taskbar compaiono dei pulsanti con icona e nome delle varie prospettive aperte e basterà selezionarne uno per aprire la prospettiva corrispondente nel workbench. Volendo è possibile non visualizzare il testo relativo al nome, per avere così più spazio a disposizione se le prospettive aperte sono molte: cliccate con il pulsante destro sulla taskbar e deselezionate la voce *Show Text*, vedrete così solamente le icone. Tornando alle varie prospettive presenti nell'installazione di base del prodotto, quattro sono specifiche per lo sviluppo Java e sono quelle su cui ci concentreremo noi: *"Java" perspective*, probabilmente quel-

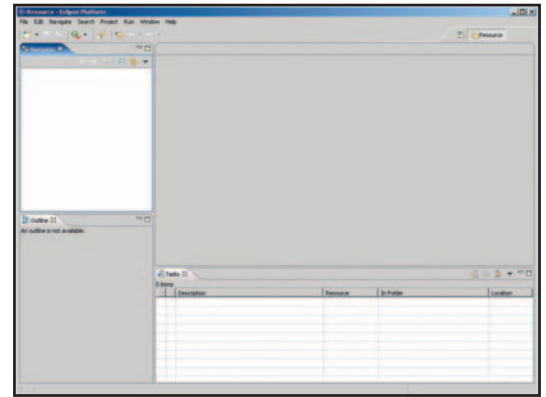



Fig. 3: **La prospettiva mostrata da Eclipse al primo lancio**

FILE DI LOG DI ECLIPSE

Se a volte Eclipse sembra comportarsi in modo strano, se avete installato qualche nuovo plug-in ma non vedete nulla di diverso, o semplicemente qualcosa non funziona come dovrebbe, andate a dare un'occhiata al file *.log* posto nella cartella del workspace: non è detto che sia tutto chiaro, ma se non altro ci rendiamo conto di qualche possibile eccezione lanciata durante l'uso del tool.

la che userete di più insieme alla *"Debug"*, *"Java Browsing"* e *"Java Type Hierarchy"*. Un'altra prospettiva per la stesura di codice Java è la *"Plug-in Development"*, di cui però non ci occuperemo in questa sede.

SVILUPPARE IN JAVA

La nostra base per le prime sperimentazioni con l'IDE Java di Eclipse sarà un semplice progetto per una calcolatrice Java (simile a quella presente tra gli accessori di Windows). Una volta presa dimestichezza con Eclipse, invece, andremo a sviluppare in varie fasi un generatore di PDF integrando librerie esterne e ne testeremo le funzionalità sotto versioni diverse del JDK, sempre dall'interno dell'IDE a nostra disposizione. Nel processo di creazione del nostro software, apprenderemo quali tool e feature Eclipse mette a disposizione di chi scrive codice in Java per facilitarne e velocizzarne il lavoro. Per cominciare, la prima cosa che dobbiamo fare è creare un progetto, in modo da avere una cartella all'interno del workspace dove tenere tutto il nostro materiale di sviluppo e dove Eclipse potrà depositare i suoi file di gestione e configurazione dell'IDE. Dal menu principale selezioniamo *File->New->Project…* e nella finestra che compare premiamo *Next* dopo aver verificato che il tipo di progetto selezionato è *"Java Project"*. Chiameremo il nostro nuovo lavoro *"JavaDevWithEclipse"* e poi premiamo *Next* di nuovo. Sarete portati alla finestra dei *"Java Settings"*, a cui potrete sempre ritornare anche dopo questo wizard cliccando con il destro sul folder del progetto nella view *"Navigation"* e selezionando *Properties* e poi *Java Build Path*. Questa finestra è molto importante in quanto ci permette di definire dei classpath aggiuntivi per la compilazione del nostro progetto e addirittura di variare il JDK con cui lavorare. Ma su questi argomenti torneremo più avanti, quando parleremo della libreria SWT e di quella PDF. Premuto il tasto *Finish*, inizia la creazione della cartella del progetto e vi viene chiesto se volete passare alla prospettiva *"Java"* (vedi Fig. 4): dite di sì ed osservate che ora abbiamo delle viste a disposizione diverse dalle precedenti. Le view che vengono offerte in questa nuova prospettiva sono indirizzate alla navigazione del codice Java (*"Package Explorer"*, *"Hierarchy"*, *"Outline"*) e alla facilitazione della stesura dello stesso (*"Problems"*, *"Javadoc"*, *"Declaration"*). Apriamo una breve parentesi perché, prima di iniziare a scrivere un programma, è buona norma configurare l'ambiente di lavoro in base alle proprie esigenze e gusti. Apriamo quindi la finestra delle opzioni (menu *Window-> Preferences*) e diamo un rapido sguardo ad alcune impostazioni di base. La finestra che ci interessa è quella della Fig. 5: alla voce *Workbench->Colors and Fonts* potete modificare carattere, dimensione e colori dei testi nei vari editor offerti, Java Editor incluso. Se passate invece a *Java->Editor,* alla pagina *Appearance* potete aggiungere i numeri di riga (checkbox *Show line numbers*), mentre alla pagina *Syntax* potete configurare la colorazione della sintassi in base alle vostre preferenze: tenete conto che nella versione 3 di Eclipse, rispetto alla precedente, potete utilizzare anche il corsivo come elemento di distacco visuale ed avete inoltre alcune opzioni in più che si attivano spuntando la checkbox *Enable advanced highlighting*. Queste includono, fra le varie possibilità, la differenziazione tra elementi statici e non, campi di una classe e variabili locali, variabili final (costanti) e non, chiamate a metodi normali e metodi astratti. Per una maggiore produttività nella stesura del codice, Eclipse offre il *"code assist"* al pari di Visual Studio .NET, Delphi e gli altri principali IDE di sviluppo del momento. Tale funzionalità viene invocata ad opera della combinazione *CTRL+Spazio* e mostra l'elenco di variabili e metodi disponibili nel contesto Java in cui si sta scrivendo. Oltre a ciò, la stessa combinazione propone anche una serie di template di codice configurabili dall'utente alla pagina *Editor->Templates* sempre nella finestra delle preferenze: qui potete modificare le voci già esistenti in base al vostro stile e aggiungerne eventualmente delle nuove. Nell'editor sarà sufficiente digitare il nome del template e premere *CTRL+ Spazio* per inserire il blocco di codice corrispondente. Altra forma di template riguarda invece la creazione di file, tipi, metodi, etc. La sezione delle preferenze interessata in questo caso è *Java->Code Style->Code Templates* e qui potete definire per esempio la struttura dei nuovi file Java

NOTA

COME SPOSTARE LE VIEW

Per spostare e sistemare in maniera differente le view di Eclipse, trattale come semplici finestre di Windows: drag and drop dalla barra del titolo. Notate che il cursore cambia forma a seconda di dove andrà a parare la view ed in che modo: se finirà ancorata ad uno dei bordi del workbench il cursore diventa una freccetta nera, se la view si aggiungerà ai tab di una serie di altre viste il cursore prende la forma di un folder.

Fig. 4: La prospettiva offerta per la stesura e lettura del codice Java

Fig. 5: La forbita finestra delle preferenze di Eclipse

che vengono creati automaticamente con Eclipse. Modificate il commento iniziale per includere il vostro nome ed ogni altra informazione che desideriate vedere sempre ogni volta che lavorate su un nuovo file. Sia in questi template che in quelli del code assist, avete a disposizione alcune variabili di ambiente che vengono rimpiazzate al volo con i valori corrispondenti (data ed ora della creazione, nome del file, del metodo, etc). Potete usare il pulsante *Insert Variable…* dell'editor di template che la combinazione *CTRL+Spazio:* entrambi mostreranno il pop-up di Fig. 6.

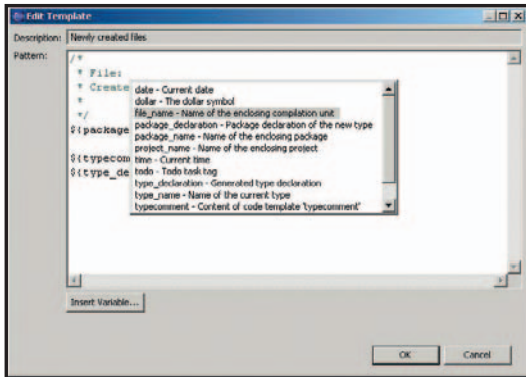



Fig. 6: **Code assist anche nella definizione dei template di codice**

PROGRAMMAZIONE VISUALE

Prima di procedere con la creazione della nostra calcolatrice, assicuratevi di avere installato il plug-in VE da *Eclipse.Org* insieme a tutte le sue dipendenze (ne abbiamo parlato nell'articolo a pag. XX). La creazione di una classe Java è una procedura molto semplice e non necessita di grandi commenti: selezionate il progetto Java con il tasto destro e dal menu contestuale selezionate *New->Visual Class.* Compilate poi il campo del nome del package (*fedmest.eclipse.calculator*) e della classe (*JCalculator*). Questa classe, dovendo fungere da finestra principale, deriverà da *JFrame* (utilizzeremo *Swing* per ora) per cui selezionate *Swing->Frame* come *Style,* spuntate la checkbox per la creazione del metodo *main* e cliccate su *Finish.* Nel *"Package Explorer"* dovreste adesso vedere il pacchetto appena definito con dentro la classe creata, come in Fig. 7. Verrà anche aperto l'editor Visuale posizionato sulla nuova classe: si tratta di un potente strumento di scrittura codice e preparazione di interfacce grafiche a cui non manca proprio nulla, al punto di fare quasi invidia anche ad altri strumenti simili di certificata qualità quali Visual Studio .NET o Delphi. Per default, l'editor mostra codice e interfaccia su una schermata sola: io preferisco invece avere due tab separati, cosa che si imposta nelle preferenze *Window->Preferences* alla pagina *Java->Visual Editor* selezionando il radio button *On separate notebook tabs.* Il risultato lo vedete in Fig. 8. La creazione visuale dei form è basata sul click and drop (nel senso che cliccate sulla palette per selezionare e poi ricliccate sul form per posizionare il controllo scelto) e come noterete dalla palette dei controlli, avete una scelta molto ampia, dai classici *label, textbox, button,* ai più sofisticati *progress bar, slider, tree control:* non manca proprio nulla! Durante l'uso di VE, inoltre, due view aggiuntive vi sono di aiuto: la view *"Properties",* che visualizza e permette di editare le proprietà dei vari elementi selezionati (dimensioni, colore, testo, etc), e la view *"Java Beans",* che offre un outline della vostra composizione grafica. Nell'esempio allegato all'articolo relativo alla calcolatrice in Java, io ho scelto un layout di tipo *GridBagLayout* (impostandolo tramite la view delle proprietà alla voce *layout* sul pannello interno del JFrame, che a sua volta raggiungete nella view *"Java Beans"* sotto il nome *jContentPane*), e VE mi permette di inserire gli elementi mostrando la posizione della griglia in cui cade il componente e l'eventuale spostamento degli altri elementi sul form in colore giallo (vedete la Fig. 9). L'utilizzo del tool è abbastanza intuitivo, soprattutto per chi già ha programmato in ambienti quali Delphi, Visual Basic o Visual Studio .NET, visto che le modalità di funzionamento sono molto simili. Anche qui possiamo, a partire da un controllo su una finestra, generare l'evento corrispondente pronto per la codifica e nel prossimo numero partiremo esattamente da lì per completare la calcolatrice ed iniziare il lavoro sul nostro secondo progetto. Per questa volta termino qui lasciandovi anche un po' di tempo per le vostre sperimentazioni. Ma abbiamo appena spolverato la superficie, perché – soprattutto per quanto riguarda gli strumenti di manipolazione del codice scritto – il nostro Eclipse ha ancora molto da offrire per tutti. Rimanete sintonizzati su queste pagine per scoprire altre caratteristiche e funzionalità!

Federico Mestrone

Fig. 8: **L'editor visuale con codice ed interfaccia su due tab separati**

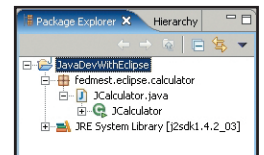



Fig. 7: **Il "Package Explorer" con la nostra nuova classe visuale**

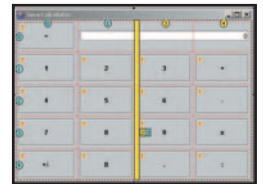



Fig. 9: **Una vista del plug-in VE alle prese con un GridBagLayout**

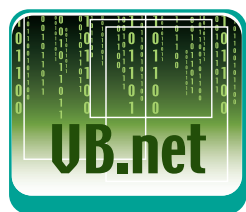
Fig. 10: **La calcolatrice esempio dell'uso di VE**

DOMANDE?

Se avete bisogno di qualche aiuto, potete scrivermi a
federico.mestrone@ioprogrammo.it
Anche se la risposta può farsi attendere molto, soprattutto quando sono sotto stress, normalmente arriva sempre!

Il namespace System.IO

Gestione di file

In questo appuntamento descriveremo le classi messe a disposizione da VB.NET per la gestione degli elementi del file system. Il codice illustrato potrà essere utilizzato nelle vostre applicazioni

Il namespace *System.IO* comprende una libreria di classi che offrono proprietà, metodi ed eventi per creare, copiare, spostare ed eliminare file e directory. Le classi maggiormente utilizzate per la gestione di file e directory sono:

- La classe **Directory** che fornisce metodi condivisi per la gestione delle directory.
- La classe **File** che fornisce metodi condivisi per la gestione dei file.
- La classe **DirectoryInfo** che rappresenta una directory specifica e, come la classe *Directory,* permette di gestire cartelle (fornisce solo metodi di istanza).
- La classe **Fileinfo** che rappresenta un file specifico e, come la classe *File,* permette di gestire file (fornisce solo metodi di istanza)
- La classe **Path** che fornisce metodi condivisi per la gestione delle informazioni sul percorso.

Per mantenere il codice il più compatto possibile, si può utilizzare l'istruzione *Imports* che semplifica l'accesso alle classi, eliminando la necessità di digitare in modo esplicito il nome completo del namespace che le contiene. Le istruzioni *Imports* devono sempre essere scritte nella parte superiore del file nel quale volete utilizzarle, prima di qualunque altro codice. Per queste ragioni, in tutti gli esempi di codice, assumeremo l'inserimento della seguente istruzione *Imports: Imports System.IO*

LA CLASSE DIRECTORY

La classe *Directory* consente di gestire le directory, permettendone la creazione, modifica e cancellazione. Fornisce, inoltre, alcuni metodi per le unità logiche del sistema *(drive)*. I metodi della classe *Directory* sono metodi statici pertanto non è necessario creare un'istanza (la classe non fornisce un costruttore), di norma i più utilizzati sono:

CreateDirectory(percorso). Permette di creare la directory indicata nell'argomento percorso. Se *percorso*, contiene delle directory intermedie che non esistono, allora vengono create anch'esse.
Delete(percorso). Permette di eliminare una directory ed il relativo contenuto
Exists(percorso). Permette di stabilire se il percorso, indicato come argomento, esiste sul disco, in questo caso restituisce il valore *True*.
GetDirectories(percorso). Permette di recuperare tutte le sottodirectory contenute nella directory corrente, restituendole in una matrice di stringhe.
GetLogicalDrives. Permette di recuperare i nomi delle unità logiche presenti nel computer restituendoli in una matrice di stringhe.
GetFiles(percorso). Permette di recuperare tutti i file contenuti in una directory, eventualmente filtrati secondo un criterio specifico, restituendoli in una matrice di stringhe.
GetParent(percorso). Permette di recuperare la directory padre del percorso specificato
Move(percorsoSorgente, percorsoDestinazione). Permette di spostare un oggetto *DirectoryInfo* ed il suo contenuto in un nuovo percorso
SetCurrentDirectory e **GetCurrentDirectory.** Permettono di impostare o recuperare la directory corrente.

Utilizzando soltanto la classe Directory ed i suoi metodi, possiamo realizzare una semplice applicazione che ci permetta di navigare nel file system.
Sulla finestra dovremo disegnare:

- Un ComboBox, dal nome *ComboDrive,* in cui visualizzeremo le unità logiche *(drive)* presenti nel sistema
- Una ListBox, dal nome *ListBoxDir,* in cui visualizzeremo le directory dell'unità logica selezionata in *ComboDrive*
- Una ListBox, dal nome *ListBoxFile,* in cui visualizzeremo i file contenuti nella directory selezionata in *ListBoxDir*

Le unità logiche dovranno essere mostrate appena viene visualizzata la finestra, pertanto nell'evento *Load* della form possiamo scrivere:

```
Private Sub Form1_Load(ByVal sender As System.Object,
        ByVal e As System.EventArgs) Handles MyBase.Load
        Dim MatStrDrive() As String
        Dim StrDrive As String
        MatStrDrive = Directory.GetLogicalDrives
        For Each StrDrive In MatStrDrive
            ComboDrive.Items.Add(StrDrive)
        Next
        ComboDrive.SelectedIndex = 1
End Sub
```

Nelle prime due righe si dovranno definire le variabili deputate a contenere la matrice dei drive ed il drive da aggiungere al ComboBox. Chiamando il metodo *GetLogicalDrives* valorizziamo la matrice dei drive. Con un ciclo *For Each… Next* cicliamo sulla matrice dei drive ed aggiungiamo ogni elemento in *ComboDrive,* con la sintassi che ormai dovremo ben conoscere. Infine mostriamo l'unità *C:* che dovrebbe avere indice uno (la prima dovrebbe essere *A:\).* Nel momento in cui l'utente seleziona un drive, dobbiamo mostrare le directory contenute nel drive selezionato, pertanto l'evento che dovrà contenere il codice necessario sarà *SelectedIndexChanged* del ComboBox

```
Private Sub ComboDrive_SelectedIndexChanged(ByVal
    sender As System.Object, ByVal e As System.EventArgs)
                    Handles ComboDrive.SelectedIndexChanged
        Dim MatStrDir() As String
        Dim StrDir As String
        Try
            MatStrDir = Directory.GetDirectories(
                            ComboDrive.SelectedItem)
            ListBoxDir.Items.Clear()
            For Each StrDir In MatStrDir
                ListBoxDir.Items.Add(StrDir)
            Next
        Catch
MessageBox.Show("Driver non pronto")
        End Try
End Sub
```

Dopo la solita dichiarazione di variabili utilizziamo il metodo *GetDirectories,* passando come argomento il drive selezionato nel ComboBox, per popolare la matrice delle directory. Utilizziamo il metodo *Clear* per pulire la ListBox ed infine con un ciclo *For Each… Next* cicliamo sulla matrice delle directory ed aggiungiamo ogni elemento nella ListBox. Il gestore di errori *Try..Catch* si rende necessario per gestire il caso di drive non pronto, che si verifica, ad esempio, quando si seleziona l'unità *A:* e nessun dischetto è stato inserito nel floppy disk. Infine, nel momento in cui l'utente seleziona una directory, dobbiamo mostrare i file contenuti nella directory selezionata, pertanto l'evento che dovrà contenere il codice necessario sarà l'evento *SelectedIndexChanged* della prima ListBox. Utilizziamo il metodo *GetFiles* passando come argomento la directory selezionata nella ListBox, per popolare la matrice dei file, ed utilizziamo la stessa struttura di codice scritta in precedenza che, per questo motivo, evitiamo di descrivere nei dettagli.

```
Private Sub ListBoxDir_SelectedIndexChanged(ByVal
    sender As System.Object, ByVal e As System.EventArgs)
                    Handles ListBoxDir.SelectedIndexChanged
        Dim MatStrFile() As String
        Dim StrFile As String
        MatStrFile = Directory.GetFiles(
                            ListBoxDir.SelectedItem)
        ListBoxFile.Items.Clear()
        For Each StrFile In MatStrFile
            ListBoxFile.Items.Add(StrFile)
        Next
End Sub
```

LA CLASSE FILE

La classe *File* fornisce i metodi per elaborare file, permettendone la cancellazione, la copia o lo spostamento. Sono, inoltre, disponibili metodi che consentono di estrarre informazioni, come la data di creazione e di modifica dei file e gli attributi; e metodi che consentono di aprire un file e creare un oggetto *FileStream* (che descriveremo nel prossimo articolo) utilizzato per le operazioni di lettura e scrittura su file. Anche i metodi della classe *File* sono metodi statici, vediamone alcuni:

Fig. 1: In figura la finestra con i controlli

I metodi **SetAttributes** e **GetAttributes** permettono di impostare o di prelevare, un valore codificato in bit, che rappresenta una combinazione dei valori possibili degli attributi di un file (*Normal, Archive, ReadOnly, Hidden,* ecc)

I metodi **SetCreationTime, SetLastAccessTime** e **SetLastWriteTime** permettono di modificare gli attributi relativi a data ed ora inerenti, rispettivamente, alla creazione, ultimo accesso e ultima scrittura di un file o una directory

Fig. 2: L'applicazione in esecuzione

I metodi **GetCreationTime, GetLastAccessTime, GetLastWriteTime** restituiscono la data e l'ora inerenti, rispettivamente, alla creazione, ultimo accesso e ultima scrittura di un file.

A questo punto possiamo migliorare l'applicazione

NOTA

La classe Directory non espone un metodo *GetAttribute*, per questo non è possibile estrarre gli attributi per una directory utilizzando tale classe. Per ovviare al problema, si può utilizzare il metodo *File.GetAttributes* che può essere applicato anche alle directory. Un'altra possibilità consiste nell'utilizzare la proprietà *Attributes* di un oggetto *DirectoryInfo.*

Nei metodi e campi che accettano un percorso, è possibile fare riferimento ad un file o solo ad una directory. Il percorso può anche fare riferimento ad un percorso relativo, un percorso assoluto, ad un server e ad un nome di condivisione. Tutti i seguenti percorsi, ad esempio, sono accettabili: "C:\MiaDir\ MioFile.txt" "C:\MiaDir" "\\MioServer\MiaCondivisione"

precedente mostrando a video le proprietà del file selezionato nella ListBox: *ListBoxFile*. L'evento che dovrà contenere il codice necessario sarà l'evento *SelectedIndexChanged* della ListBox. Molto semplicemente, dovremo utilizzare i metodi *GetCreationTime, GetLastAccessTime, GetLastWriteTime,* passando come argomento il file selezionato nella ListBox, e comporre il messaggio da far comparire a video con il classico *MessageBox. Show*

```
Private Sub ListBoxFile_SelectedIndexChanged(ByVal
    sender As System.Object, ByVal e As System.EventArgs)
                        Handles ListBoxFile.SelectedIndexChanged
        Dim DataCreazione As Date
        Dim DataModifica As Date
        Dim DataAccesso As Date
        DataCreazione = File.GetCreationTime(
                                        ListBoxFile.SelectedItem)
        DataModifica = File.GetLastWriteTime(
                                        ListBoxFile.SelectedItem)
        DataAccesso = File.GetLastAccessTime(
                                        ListBoxFile.SelectedItem)
        Dim Messaggio As String
        Messaggio = "File " & ListBoxFile.SelectedItem
                                                        & vbCrLf
        Messaggio = Messaggio & "Data di Creazione :"
                                        & DataCreazione & vbCrLf
        Messaggio = Messaggio & "Data Ultima Modifica:
                                        " & DataModifica & vbCrLf
        Messaggio = Messaggio & "Data Ultimo Accesso:
                                        " & DataAccesso & vbCrLf
        MessageBox.Show(Messaggio)
End Sub
```

Per le operazioni di lettura e scrittura su un file, si deve utilizzare il metodo *Open*. Il metodo *Open* apre un oggetto *FileStream* (che tratteremo dettagliatamente nel prossimo articolo), nel percorso specificato e con le modalità di accesso passate come argomento. Si devono impostare tre valori:

Fig. 3: *Il messaggio visualizzato dall'applicazione*

FileMode permette di indicare le modalità di apertura di un file. Può assumere i valori: *Append, Create, CreateNew, Open, OpenOrCreate* e *Truncate*. Le modalità *Open* e *Append* falliscono nel caso il file non esista; *Create* o *CreateNew* falliscono se il file esiste già. Per aprire un file esistente si deve utilizzare *Open*. Per aprire un file o crearne uno se ancora non esiste si deve utilizzare *OpenOrCreate*. Per aggiungere un elemento ad un file, si deve utilizzare *Append*
FileAccess permette di indicare le modalità di accesso in lettura e scrittura di un file. Può assumere i valori; *Read, Write, ReadWrite*.
FileShare permette di indicare il tipo di accesso al file di cui dispongono altri thread aperti. Può assumere i valori *None* (sono vietate tutte le operazioni), *ReadWrite* (sono consentite tutte le operazioni), *Read, Write* e *Inheritable*.

Esistono inoltre tre varianti del metodo *Open* che restituiscono un oggetto *FileStream:*

- **Create.** Crea un file nel percorso completo specificato come argomento.
- **OpenRead.** Apre un file esistente per la lettura.
- **OpenWrite.** Apre un file esistente per la scrittura.

Infine, sono disponibili tre metodi che riguardano specificamente i file di testo: *CreateText, OpenText* e *AppendText* e si possono utilizzare per creare o aprire un file di testo o per aggiungere del testo in un file. Questi tre metodi restituiscono un oggetto *StreamReader* o un oggetto *StreamWriter* che saranno descritti nel prossimo articolo.

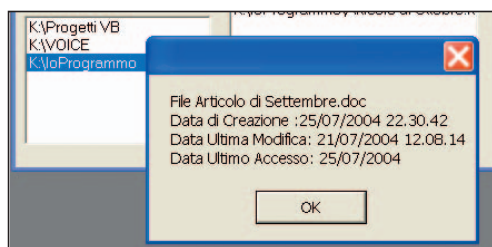
LA CLASSE DIRECTORYINFO

La classe *DirectoryInfo* rappresenta una singola directory. Come la classe *FileInfo,* descritta in seguito, deriva dalla classe virtuale *FileSystemInfo* e, pertanto, dispone di numerose proprietà e metodi in comune come:

- **Name** che restituisce il nome della directory (o file per la classe *FileInfo*).
- **FullName** che restituisce il nome completo della directory (o file per la classe *FileInfo*), incluso il relativo percorso.
- **Attributes** che permette di impostare od ottenere gli attributi della directory (o file per la classe *FileInfo*).
- **CreationTime, LastWriteTime** e **LastAccessTime** che permettono di impostare od ottenere la data e l'ora: della creazione, dell'ultima modifica o dell'ultimo accesso della directory o file

A differenza delle classi descritte finora, non è possibile utilizzare la classe *DirectoryInfo* ed i suoi metodi senza aver ottenuto un riferimento ad un oggetto di tipo *DirectoryInfo.* Per questo si deve adoperare il relativo metodo costruttore:

```
Public Sub New( ByVal Percorso As String)
```

La classe *DirectoryInfo* fornisce molti metodi analoghi a quelli della classe *Directory,* con la differenza che restituiscono degli oggetti *FileSystemInfo,* piuttosto che semplici stringhe. Possiamo ora riscrivere il codice dell'evento *SelectedIndexChanged* di *ComboDrive* utilizzando la classe *DirectoryInfo.*

```
Private Sub ComboDrive_SelectedIndexChanged(ByVal
    sender As System.Object, ByVal e As System.EventArgs)
                Handles ComboDrive.SelectedIndexChanged
        Dim MatFsiDir() As FileSystemInfo
        Dim FsiDir As FileSystemInfo
        Try
            ListBoxDir.Items.Clear()
            MatFsiDir = New DirectoryInfo(
                ComboDrive.SelectedItem).GetDirectories
            For Each FsiDir In MatFsiDir
                ListBoxDir.Items.Add(FsiDir.FullName)
            Next
        Catch
            MessageBox.Show("Driver non pronto")
        End Try
End Sub
```

La prima differenza che risulta subito evidente è il tipo delle due variabili che non è più *String* ma *FileSystemInfo*. Per valorizzare la matrice di oggetti *FileSystemInfo* è stato utilizzato il costruttore della classe *DirectoryInfo*, passando come argomento il drive selezionato nel ComboBox ed utilizzando l'analogo metodo *GetDirectories*.
L'ultima differenza è nel modo in cui è stato riempito il ListBox, è infatti necessario utilizzare il metodo *FullName* per visualizzare il percorso completo delle sottodirectory.

LA CLASSE FILEINFO

La classe *FileInfo* rappresenta un singolo file. Analogamente a quanto detto per la classe *DirectoryInfo*, non è possibile utilizzare la classe *FileInfo* ed i suoi metodi, senza aver ottenuto un riferimento ad un oggetto di tipo *FileInfo*, per questo si deve adoperare il relativo metodo costruttore dove specificare il nome completo del file:

```
Public Sub New( ByVal NomeFile As String)
```

Torniamo alla nostra applicazione e riscriviamo il codice, che mostra a video le proprietà del file selezionato nella ListBox, utilizzando la classe *FileInfo*.

```
Private Sub ListBoxFile_SelectedIndexChanged(ByVal
    sender As System.Object, ByVal e As System.EventArgs)
                Handles ListBoxFile.SelectedIndexChanged
        Dim DataCreazione As Date
        Dim DataModifica As Date
        Dim DataAccesso As Date
        Dim FsiFile As FileSystemInfo
        FsiFile = New FileInfo(ListBoxFile.SelectedItem)
        DataCreazione = FsiFile.CreationTime
        DataModifica = FsiFile.LastWriteTime
        DataAccesso = FsiFile.LastAccessTime
        Dim Messaggio As String
        Messaggio = "File " & FsiFile.Name & vbCrLf
        Messaggio = Messaggio & "Data di Creazione :"
                        & DataCreazione & vbCrLf
        Messaggio = Messaggio & "Data Ultima Modifica:
                        " & DataModifica & vbCrLf
        Messaggio = Messaggio & "Data Ultimo Accesso:
                        " & DataAccesso & vbCrLf
        MessageBox.Show(Messaggio)
End Sub
```

Anche in questo caso la variabile non è più di tipo *String* ma di tipo *FileSystemInfo*. Per valorizzare la variabile è stato utilizzato il costruttore della classe *FileInfo* passando come argomento il nome del file selezionato nel ListBox. Infine, per comporre il messaggio riassuntivo, delle proprietà del file selezionato, sono state utilizzate le proprietà *CreationTime*, *LastWriteTime* e *LastAccessTime*.

LA CLASSE PATH

La classe *Path* espone campi e metodi che consentono di ottenere informazioni su percorsi di file e directory. La classe *Path* espone cinque campi statici che permettono di ottenere informazioni sui drive validi e sui separatori leciti per i nomi dei file, nel sistema operativo corrente.

AltDirectorySeparatorChar restituisce un carattere alternativo utilizzato per separare i livelli di directory.

- *Path.AltDirectorySeparatorChar* restituisce /

DirectorySeparatorChar restituisce il carattere utilizzato per separare i livelli di directory

- *Path.DirectorySeparatorChar* restituisce \

InvalidPathChars restituisce una matrice di caratteri che non possono essere utilizzati nella definizione di un percorso.

- *Path.InvalidPathChars* restituisce "<>|

PathSeparator restituisce il carattere separatore, usato per separare le stringhe di percorso nelle variabili di ambiente.

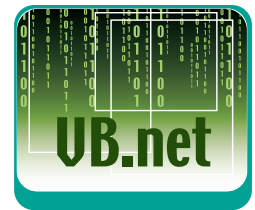
- *Path.PathSeparator* restituisce ;

VolumeSeparatorChar restituisce il carattere separatore di volume.

- *Path.VolumeSeparatorChar* restituisce :

CONCLUSIONI

In questo articolo abbiamo descritto le classi contenute nel namespace *System.IO* che permettono di lavorare con gli elementi del filesystem (directory, file, drive). Nel prossimo articolo ci occuperemo delle classi del namespace *System.IO* che permettono di leggere e scrivere un file.

Luigi Buono

ECCEZIONI DI FILE.LOAD

SecurityException **non si dispone dell'autorizzazione necessaria.**

ArgumentException **il percorso specificato è una stringa di lunghezza zero, contiene solo spazi vuoti oppure uno o più caratteri non validi definiti da *InvalidPathChars*.**

ArgumentNullException **il percorso è *Nothing*.**

PathTooLongException **il percorso o le informazioni sul percorso assoluto superano la lunghezza massima definita dal sistema.**

DirectoryNotFoundException **La directory specificata non esiste.**

UnauthorizedAccessException **il percorso specificato fa riferimento ad un file che è in sola lettura e access non è *Read*. Si verifica anche nel caso in cui il percorso è una directory.**

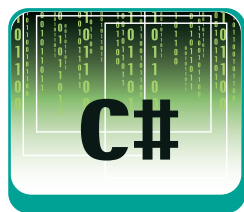
ArgumentOutOfRangeException ***mode*** **o** ***access*** **o** ***share*** **ha specificato un valore non valido.**

FileNotFoundException **Il file specificato non esiste.**

NotSupportedException **il percorso contiene il carattere due punti *(:)* all'interno della stringa.**

Il preprocessore per migliorare il debug del software

Il Debug con il preprocessore

Il preprocessore è uno strumento che C# eredita direttamente da C++. Ne illustreremo le funzionalità adattandone l'utilizzo alle fasi di debug del software

C# dispone di un comodo preprocessore. Ok, ma che cos'è un preprocessore? Il preprocessore è, in termini semplici, uno strumento che si interpone tra il codice scritto dal programmatore ed il lavoro del compilatore. In pratica, quando lanciamo il comando di compilazione *csc* su un sorgente, il primo strumento di analisi ed interpretazione incaricato di esaminare il nostro lavoro è proprio il preprocessore. Il preprocessore scorre tutto il codice alla ricerca di speciali comandi, detti direttive per il preprocessore. Ciascuna direttiva viene interpretata e valutata. L'output del preprocessore è costituito dal nostro codice sorgente opportunamente modificato, là dove le nostre direttive sono state sostituite dal risultato della corrispondente interpretazione. Questo codice risultante viene poi fornito al compilatore, per la vera e propria generazione del codice oggetto. Quindi, sfruttando le possibilità offerte dal preprocessore, è possibile rendere dinamico il sorgente di un software, facendo in modo che la compilazione percorra una strada piuttosto che un'altra, secondo alcune condizioni stabilite a priori. Credete che sia tutta teoria e che il preprocessore non serva a nulla?
Continuate a leggere!

C code
Preprocessor
Expanded C program
Compiler
object file

Fig. 1: *I passaggi che conducono dal sorgente al compilato*

LE DIRETTIVE PER IL PREPROCESSORE

In C# il preprocessore, con le sue direttive, è un chiaro retaggio di C++. Benché in un linguaggio strettamente orientato agli oggetti come C# il preprocessore non giochi più un ruolo fondamentale (come era nei linguaggi di una volta) spesso alcune direttive tornano utili per mettere in piedi una sorta di compilazione condizionale. Il momento in cui questa possibilità più si fa apprezzare è a tempo di debug, come vedremo a breve. La seguente tabella ci mostra le direttive previste dal preprocessore di C#. Chi non è estraneo a C++ ne riconoscerà più di una:

#define	#elif	#else	#endif
#endregion	#error	#if	#line
#region	#undef	#warning	

Come è possibile osservare ogni direttiva per il preprocessore inizia con il simbolo cancelletto *(#)*. Inoltre è importate sapere, a questo punto, che ciascuna direttiva deve essere compresa in una sola riga, e che in una riga può essere introdotta una sola direttiva, senza altri elementi. Insomma: ciascuna direttiva deve essere gestita in una sola e dedicata linea del codice.
Nei prossimi paragrafi saranno esaminate le

direttive più interessanti fra quelle appena presentate.

LA DIRETTIVA #DEFINE

Cominciamo la nostra analisi dalla direttiva *#define*. Essa è utile per definire un *simbolo*. Un *simbolo* è una sequenza di caratteri.
La forma generale della direttiva è:

```
#define SIMBOLO
```

Inquadriamo la situazione sotto il seguente punto di vista: il preprocessore interpreta un proprio linguaggio, alla base del quale stanno le direttive sopra mostrate. I *simboli*, in questa speciale codifica, sono come delle variabili. Non si tratta, però, di variabili flessibili come quelle di C#, all'interno delle quali è possibile accumulare valori dei più disparati tipi: un *simbolo* è una sorta di variabile booleana, che può valere soltanto *true* oppure *false*. Quando un *simbolo* è *definito*, continuando l'analogia, il suo valore è *true*; quando non è *definito*, invece, per il preprocessore il simbolo ha valore *false*.

LE DIRETTIVE #IF E #ENDIF

La direttiva *#if* è alla base della già citata compilazione condizionale. La sua forma di utilizzo è la seguente:

```
#if condizione
  // Codice C#
#endif
```

Se la condizione espressa è vera, il blocco di codice compreso tra *#if* e *#endif* sarà incluso nel sorgente sottoposto al compilatore. In caso contrario, ovviamente, lo stesso codice sarà escluso dall'input fornito al compilatore, proprio come se il programmatore non lo avesse mai scritto. Già, ma come si esprimono delle condizioni corrette per la direttiva *#if*?
Giusto nel paragrafo precedente è stato spiegato come i simboli siano una sorta di variabili booleane, che possono essere semplicemente definiti oppure non definiti. Ecco allora una prima semplice maniera per la composizione di condizioni valide:

```
#if SIMBOLO
  // Codice C#
#endif
```

Se il simbolo richiamato è stato precedentemente definito attraverso un'opportuna istruzione *#define*, allora il codice C# compreso nella struttura entrerà a far parte del sorgente da fornire al compilatore.
Gli elementi esaminati ci permettono di comporre un primo esempio:

```
#define PROVA
using System;
class Test
  { public static void Main()
    { Console.WriteLine("Questo si vede sempre.");
      #if PROVA
      Console.WriteLine("Questo si vede solo se PROVA
                                   è un simbolo definito.");
      #endif
    }
}
```

Il simbolo *PROVA* viene definito in apertura. Per questo motivo la condizione valutata alla riga

```
#if PROVA
```

risulterà sempre verificata. Di conseguenza l'istruzione

```
Console.WriteLine("Questo si vede solo se PROVA e' un
                                   simbolo definito.");
```

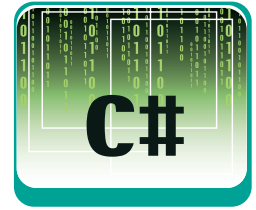
farà parte del codice compilato. Avviando il risultato della compilazione si leggerà in output:

Questo si vede sempre.
Questo si vede solo se PROVA e' un simbolo definito.

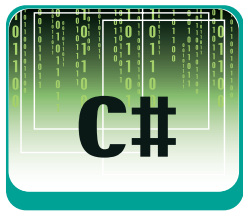
Provate ora a rimuovere dal sorgente la definizione del simbolo *PROVA*. Fate così:

```
// #define PROVA - Questo ora è un semplice com-
mento
using System;
class Test
  { public static void Main()
    { Console.WriteLine("Questo si vede sempre.");
      #if PROVA
      Console.WriteLine("Questo si vede solo se PROVA
                                   è un simbolo definito.");
      #endif
    }
}
```

Ricompilate il tutto e avviate l'eseguibile generato. Questa volta l'output sarà soltanto:

Questo si vede sempre.

- **GUIDA A C#**
  Herbert Schildt
  (McGraw-Hill)
  ISBN 88-386-4264-8
  2002

- **INTRODUZIONE A C#**
  Eric Gunnerson
  (Mondadori Informatica)
  ISBN 88-8331-185-X
  2001

- **C# GUIDA PER LO SVILUPPATORE**
  Simon Robinson e altri
  (Hoepli)
  ISBN 88-203-2962-X
  2001

Prima domanda che spesso pone, a questo punto del discorso, chi è nuovo all'utilizzo del preprocessore e ancora deve chiarirsi bene le idee sui meccanismi del suo funzionamento: non è forse lo stesso risultato che si ottiene con un codice come il seguente, senza ricorrere al preprocessore e alle sue direttive?

```
using System;
class Test
 {  private static bool prova = true;
 public static void Main()
 {  Console.WriteLine("Questo si vede sempre.");
     if (prova)
     { Console.WriteLine("Questo si vede solo se prova
                                              e' true.");
     }
 }
}
```

All'apparenza il risultato è lo stesso. In sostanza, però, c'è una grossa differenza: il preprocessore opera prima che il compilatore entri in azione. Con un codice come quello appena mostrato, che definisce una semplice variabile booleana di C#, si sposta il problema al momento dell'esecuzione. Ogni volta che il software sarà eseguito l'ambiente di runtime dovrà verificare il valore della variabile prova, e decidere in base ad esso se eseguire o meno il blocco di codice collegato. Se prova è false il codice che non sarà mai eseguito farà parte lo stesso dell'eseguibile finale, apportando come contributo soltanto il proprio peso. Con il preprocessore non avviene nulla di tutto ciò. La definizione di un simbolo è considerata prima della compilazione. Se il simbolo *PROVA* non è definito, il codice collegato non entrerà mai a far parte dell'eseguibile finale, né l'ambiente di runtime dovrà mai valutare dei valori booleani per stabilire cosa eseguire e cosa no. Semplicemente sarà come se quel codice non l'avessimo mai scritto. Il software sarà più snello e l'esecuzione più veloce. Chiarito questo, viene la seconda domanda: ma a che serve? Certo, usando un simbolo chiamato *PROVA* non ho dato implicitamente indicazioni sull'utilità di questo meccanismo. Perciò proviamo ad esaminare un esempio analogo ai precedenti, che però al posto di *PROVA* fa uso di un simbolo chiamato *DEBUG:*

```
#define DEBUG
using System;
class Test
{  public static void Main()
   { Console.WriteLine("Questo si vede sempre.");
     // Codice del programma.
     #if DEBUG
     // Qui andrebbero delle operazioni di debug,
                              ad esempio la stampa su
     // console del contenuto di certe variabili da tenere
                                  sotto osservazione.
    Console.WriteLine("Questo si vede solo se DEBUG e'
                                  un simbolo definito.");
    #endif
  }
}
```

Vi è mai capitato di lavorare a programmi di notevoli dimensioni e, ad un certo punto, di riscontrare dei problemi di cui non siete in grado di individuare al volo l'origine? In casi come questo la miglior soluzione possibile è un po' di sano debug, usando del codice appositamente studiato per verificare variabili e condizioni dalle quali il problema potrebbe dipendere. Grazie al preprocessore potrete scrivere tutto il codice di debug che desiderate. Finché vi sarà utile fare il debug, lascerete definito il simbolo corrispondente. Nel momento in cui vorrete distribuire il vostro lavoro, vi sarà sufficiente annullare la definizione del simbolo, e sarà come se tutto il vostro codice di debug non fosse mai esistito. All'interno dei sorgenti però, questo codice continuerà ad esistere, ed un domani che tornerete a mettere mano al software potrete di nuovo farne uso reinserendo la definizione di un semplice simbolo. Ecco qui spiegato il principale scopo del preprocessore di C#. Dopo questa digressione, torniamo all'analisi della direttiva *#if* e delle condizioni ad essa associate. La forma condizionale esaminata sinora, basata sulla verifica della definizione di un solo simbolo, è la più semplice fra quelle disponibili. Il preprocessore comprende anche gli operatori logici *&& (and), || (or)* e *! (not)*. Inoltre, le parentesi tonde possono essere utilizzate per esprimere condizioni di maggiore complessità. In questo modo è possibile descrivere condizioni basate sulla verifica della definizione di più di un simbolo, come nel seguente esempio:

```
#define DEBUG1
#define DEBUG2
using System;
class Test
{  public static void Main()
   { Console.WriteLine("Questo si vede sempre.");
     #if DEBUG1
     Console.WriteLine("Solo se DEBUG1 e' definito.");
     #endif
     #if !DEBUG1
     Console.WriteLine("Solo se DEBUG1 non e' definito.");
     #endif
     #if DEBUG2
     Console.WriteLine("Solo se DEBUG2 e' definito.");
```

```
      #endif
      #if !DEBUG2
      Console.WriteLine("Solo se DEBUG2 non e' definito.");
    #endif
    #if DEBUG1 && DEBUG2
    Console.WriteLine("Solo se DEBUG1 e DEBUG2 sono
                                        entrambi definiti.");
    #endif
    #if DEBUG1 || DEBUG2
    Console.WriteLine("Solo se almeno uno tra DEBUG1
                                      e DEBUG2 è definito.");
    #endif
  }
}
```

Divertitevi a variare le definizioni dei simboli *DEBUG1* e *DEBUG2*, ricompilate ed osservate il risultato.

LE DIRETTIVE #ELSE E #ELIF

Le strutture di compilazione condizionale realizzate per il preprocessore mediante la direttiva *#if* possono essere ulteriormente complicate servendosi delle direttive *#else* e *#elif*. Per dirlo in termini immediati, *#else* corrisponde all'istruzione *else* di C#, mentre *#elif* è l'equivalente di *else if*. Suppongo che non avrete difficoltà a comprendere il seguente esempio:

```
// #define VERSIONE_HOME
#define VERSIONE_PROFESSIONAL
// #define VERSIONE_ULTRAPROFESSIONAL
using System;
class Test
{
    public static void Main()
   {
      #if VERSIONE_HOME
      Console.WriteLine("Hai la versione Home del
                                             software");
      #elif VERSIONE_PROFESSIONAL
      Console.WriteLine("Hai la versione Professional del
                                             software");
      #elif VERSIONE_ULTRAPROFESSIONAL
      Console.WriteLine("Hai la versione UltraProfessional
                                         del software");
      #else
      Console.WriteLine("Ma che versione hai?!?");
      #endif
   }
}
```

Come al solito, potrete meglio verificare il funzionamento del codice variando le definizioni iniziali.

LA DIRETTIVA #UNDEF

Un simbolo risulta non definito in due casi:

1. quando il programmatore non lo ha mai definito servendosi della direttiva *#define*
2. dopo che una sua precedente definizione è stata esplicitamente annullata. La direttiva utile per annullare la definizione di un simbolo è *#undef*. L'utilizzo è intuitivo:

```
#undef SIMBOLO
```

LE DIRETTIVE #ERROR E #WARNING

La direttiva *#error* provoca l'interruzione della compilazione del sorgente, emettendo sullo standard output un messaggio di errore.
Ecco un esempio:

```
using System;
class Test
{  public static void Main()
   {  Console.WriteLine("Fin qui compila");
        #error Errore irreversibile! Reinstallare Windows!
        Console.WriteLine("Questo non lo compila");
   }
}
```

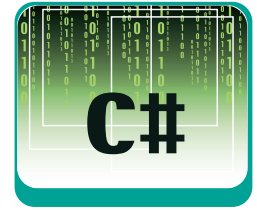
Quando si andrà a compilare il sorgente si riscontrerà un messaggio di errore del tipo:

Esempio07.cs(9,12): error CS1029: #error: "Errore irreversibile! Reinstallare Windows!"

La direttiva *#warning*, come *#error*, emette durante la compilazione un messaggio sullo standard output ma, al contrario della sua collega, non interrompe l'operazione. Il suo scopo è semplicemente quello di lanciare un avviso:

```
using System;
class Test
{
    public static void Main()
   {
      Console.WriteLine("Fin qui compila");
      #warning Aiuto! Mi stanno compilando!
      Console.WriteLine("Compila anche questo");
   }
}
```

Provate a compilare questo codice per verificare quanto asserito.

Carlo Pelliccia

Se desideri contattare l'autore di questo articolo scrivi a
carlo.pelliccia@ioprogrammo.it

Classi astratte e interfacce in Java

Il mondo delle idee

Questo mese ci solleveremo dalle piccolezze della programmazione quotidiana per entrare in un mondo nuovo: il mondo delle idee, dove gli oggetti comunicano attraverso purissime interfacce

Questo mese incontrerai un nuovo modificatore di accesso: *protected*. Conoscerai le classi astratte e le interfacce. Imparerai un modo nuovo di usare la parola *final*.

COMINCIAMO!

Signore e signori, vi presento la classe *Cantante:*

```
public class Cantante {
   public void canta() {} }
```

Questa classe è fatta apposta per essere ereditata. Il metodo *canta()* non fa niente, ma le sottoclassi di *Cantante* possono ridefinirlo per fare qualcosa di utile. Ad esempio:

```
public class NinoDAngelo extends Cantante {
   public void canta() {
      System.out.println("...'Nnu jeans e
                          'nna magliettaaa...");}}
public class Eminem extends Cantante {
   public void canta() {
      System.out.println("Will the real Slim Shady
                          please stand up?");
      insultaPubblico(); }
   private void insultaPubblico() {
      System.out.println("Take this!");} }
```

Nino D'Angelo si limita a fare una cantatina, mentre Eminem coglie anche l'occasione per insultare un po' il pubblico. Ora possiamo sfruttare il polimorfismo per far esibire questi simpatici personaggi:

```
public class Concerto {
   public static void main(String[] args) {
      Cantante c = chiamaCantanteSulPalco();
      iniziaConcerto(c);}
   private static void iniziaConcerto(Cantante c) {
      c.canta(); }
   private static Cantante chiamaCantanteSulPalco() {
      return new Eminem();}
}
```

REQUISITI

Conoscenze richieste
Concetti di ereditarietà e polimorfismo

Software
Java 2 Standard Edition SDK 1.3 o superiore.

Impegno

Tempo di realizzazione

Ho isolato la creazione del cantante nel metodo *chiamaCantanteSulPalco()*, quindi ti basta modificare questo metodo per far cantare qualcun altro. Così com'è, il programma fa esibire Eminem:

```
Will the real Slim Shady please stand up?
Take this!
```

IL MOMENTO DELLA SERIETÀ

Dopo questa esibizione di musica leggera, sento il bisogno di un po' di cultura:

```
public class CantanteLirico extends Cantante {
   public final void canta() {
      schiarisciVoce();
      produciAcuto();
      ringraziaPubblico();}
   private void schiarisciVoce() {
      System.out.println("Eh-ehm...");}
   protected void produciAcuto() {}
   private void ringraziaPubblico() {
      System.out.println("Grazie, cari, grazie...");}}
```

Un *CantanteLirico* è anche un *Cantante*, quindi potremo usare gli oggetti di questo tipo nel programma *Concerto*. Cosa ha un *CantanteLirico* in più di un normale *Cantante*? Una maggiore preparazione prima del concerto, e un miglior congedo alla fine. Ma un *CantanteLirico* (come un *Cantante*) non è pensato per essere istanziato direttamente, ma solo per essere derivato. Il metodo *canta()* del cantante lirico chiama a sua volta altri tre metodi, solo uno dei quali (*produciAcuto()*) può essere ridefinito dalle sottoclassi. Gli altri due metodi sono privati, quindi le sottoclassi non li vedono nemmeno. E se una sottoclasse decidesse di ridefinire il metodo *canta()*? Non possiamo proteggere questo metodo dichiarandolo private, perché ne abbiamo bisogno nei programmi come *Concerto* (e anche perché se ci provi riceverai un errore del compilatore, arrabbiato perché cerchi di ridurre la visibilità di un metodo definito in una

superclasse). Per evitare che le sottoclassi possano cambiare questo metodo, l'ho dichiarato *final*. Finora hai sempre visto la parola *final* applicata ai campi per segnalare che il campo ha valore costanti. La parola *final* applicata ad un metodo, invece, significa: "questo metodo non può essere ridefinito". Quindi le sottoclassi non possono fare l'override del metodo *canta()*, anche se è pubblico. L'implementazione di *canta()* nella classe *CantanteLirico* è quello che si chiama, nel gergo del design a oggetti, un metodo *template*, cioè un metodo che si limita a chiamare altri metodi e ne definisce l'ordine. Di solito alcuni tra i metodi chiamati da un metodo template sono fatti apposta per essere ridefiniti. In questo caso i metodi *schiarisciVoce()* e *ringraziaPubblico()* sono privati, quindi non possono essere ridefiniti; ma le sottoclassi possono ridefinire *produciAcuto()* per cambiare il comportamento del cantante nel momento culmine della serata. Quindi questa implementazione di *canta()* dice: "ecco cosa succederà, e in che ordine – ma per alcune di queste cose lascio che sia la mia sottoclasse a decidere i dettagli". A tempo di esecuzione il metodo *canta()* "scenderà nella sottoclasse" a chiamare una specifica implementazione di *produciAcuto()*. Questo è un altro esempio del polimorfismo in azione. Quando ho scritto il metodo *CantanteLirico.produciAcuto()* ho usato una nuova parola chiave: *protected*. *protected* è un modificatore di accesso, come *private* e *public*. Se un campo o un metodo è *protected*, allora può essere visto solo dalle sottoclassi. Questo significa che le sottoclassi di *CantanteLirico* vedono il metodo *produciAcuto()*, e possono ridefinirlo. Quindi un metodo protetto è in un certo senso "privato per la gerarchia", anziché "privato per la classe". In realtà non è proprio così, perché per qualche buffo motivo i creatori di Java hanno deciso che un metodo *protetto* è anche automaticamente "friendly", cioè visibile a tutte le classi nello stesso package. La cosa mi ha sempre lasciato perplesso, perché si tratta di due cose molto diverse. Trovo che l'interpretazione generosa che Java dà di *protected* renda questa parola chiave pressoché inutile nella gran parte dei casi. Nel nostro caso, tanto sarebbe valso dichiarare il metodo *public*, perché la classe *Concerto* (che risiede nello stesso package delle altre classi) è comunque in grado di vederlo e usarlo anche da *protected*. Ma ormai il linguaggio è così, e dobbiamo tenercelo com'è. Ho comunque usato *protected* per mandare un messaggio al lettore: questo metodo riguarda le sottoclassi, e a differenza di *canta()* non dovrebbe essere chiamato da chi non è "della famiglia".

LA NOTTE DELLE STELLE

Ecco un paio di sottoclassi di *CantanteLirico*:

```
public class Pavarotti extends CantanteLirico {
  protected void produciAcuto() {
    System.out.println("OOOOOOOOOOOOOOOOOH!"); }}
public class Callas extends CantanteLirico {
  protected void produciAcuto() {
    System.out.println("IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIH!");}}
```

Ora possiamo modificare *Concerto* per restituire un bel *Pavarotti*:

```
public class Concerto { <...>
  private static Cantante chiamaCantanteSulPalco() {
    return new Pavarotti();} }
```

Ed ecco il risultato:

Eh-ehm...
OOOOOOOOOOOOOOOOOH!
Grazie, grazie...

Tutta un'altra cosa rispetto a Eminem!

UN PO' DI CHIAREZZA

Cantante definisce il metodo *canta()*, e tutte le sottoclassi lo ridefiniscono. *CantanteLirico* definisce il metodo *produciAcuto()*, che viene ridefinito dalle sue sottoclassi. Concerto crea un particolare *Cantante* e lo usa (in UML, la freccia tratteggiata con la punta aperta definisce una generica relazione; ho specificato di quale relazione si tratta mettendo la parola *"crea"* tra apici vicino alla freccia). Non sono particolarmente entusiasta delle classi *Cantante* e *CantanteLirico*. Entrambe sono lì solo per essere ereditate, e definiscono un paio di metodi vuoti che servono solo per essere ridefiniti nelle sottoclassi. Nonostante ciò, nessuno impedisce ai client di creare un *Cantante anonimo:*

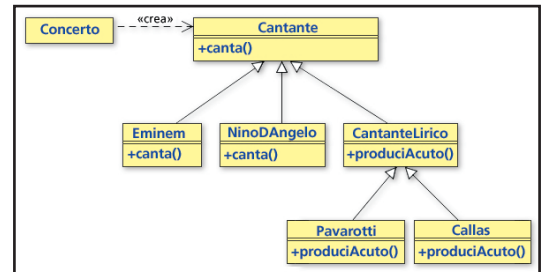
Fig. 1: **Diagramma delle classi**

```
Cantante c = new Cantante();
c.canta(); // ???
```

Cosa succede in questo caso? Nulla, perché il metodo *Cantante.()* non fa niente. Mi piacerebbe poter indicare chiaramente che nessuno dovrebbe mai chiamare direttamente il costruttore di *Cantante*. Ci sono diversi modi per ottenere questo risultato, ma il più semplice è usare la parola chiave *abstract:*

```
public abstract class Cantante {
   <...>}
```

INTERFACCE IN CAMPO

Non solo una *interface* non può contenere metodi, non può nemmeno contenere campi, a meno che questi campi non siano delle costanti (cioè *static* e *public*).

Usando quello che sai di UML, prova a fare uno schizzo delle classi che hai visto finora. Puoi vedere una soluzione in Fig. 1.

La parola chiave *abstract* dice che questa classe non può essere istanziata, cioè non puoi usarla per creare oggetti.

```
Cantante c = new Cantante();  // errore di compilazione
```

Tutto quello che puoi fare è creare un oggetto di una sottoclasse e fare un *upcast*:

```
Cantante c = new Eminem();
```

Si dice che *Cantante* è una classe astratta, mentre *Eminem* è una classe concreta. Sembrerebbe una cosa da poco, ma le classi astratte sono utili anche perché possono contenere metodi astratti. Guarda ancora *Cantante:*

```
public abstract class Cantante {
   public void canta() {} }
```

Non ti sembra stupido quel metodo *canta()* tutto solo e vuoto? Oltre ad avere un aspetto inutile è anche potenzialmente pericoloso: se scrivi una sottoclasse e ti dimentichi di ridefinire questo metodo, il tuo nuovo *Cantante* erediterà la versione vuota del metodo e risulterà completamente afono. In questo caso sarebbe facile trovare l'errore, ma in un sistema complicato questo genere di bug può essere un vero problema. Per fortuna ci viene ancora in soccorso la parola *abstract:*

```
public abstract class Cantante {
   public abstract void canta(); }
```

Ora il metodo è astratto, il che significa che diciamo che esiste (dichiariamo: un *Cantante* può cantare) e diciamo anche come è fatto (la sua signature: nome, parametri e tipo di ritorno) ma non come funziona dentro. Al posto delle parentesi graffe che definiscono il corpo del metodo abbiamo un bel punto e virgola. Se cerchi di dare un corpo concreto ad un metodo astratto, il compilatore si arrabbia. La cosa ti può sembrare strana. Come è possibile che questo metodo non contenga codice? Ricorda però che *Cantante* è una classe astratta, e quindi non verrà mai usata direttamente per creare un oggetto. Dato che un metodo astratto può esistere solo all'interno di una classe astratta, il compilatore controllerà che tutte le sottoclassi concrete di *Cantante* definiscano un metodo *public void canta()*. In pratica questo è un modo per dire: tutti i cantanti devono avere questo metodo, ma ancora non ho idea di come sarà fatto; so solo che faccia ha, e garantisco che esiste. Quindi il metodo *Concerto.iniziaConcerto()* può chiamare il metodo, anche se nessuno lo ha ancora definito. Questa faccenda delle classi astratte sembra una buona idea. Possiamo applicare lo stesso principio alla classe *CantanteLirico*. Anche *CantanteLirico* non deve essere istanziata direttamente, e contiene un “metodo segnaposto” che può diventare astratto:

```
public abstract class CantanteLirico extends Cantante {
   <...>
   protected abstract void produciAcuto();
   <...>}
```

Ricorda sempre che le sottoclassi concrete devono implementare tutti i metodi astratti dichiarati dalle loro antenate. *Pavarotti* è una classe concreta, quindi non può avere metodi astratti. Infatti non ne ha. Entrambi i metodi astratti della gerarchia hanno una loro implementazione in un *Pavarotti:* il metodo astratto *Cantante.canta()* è implementato in *CantanteLirico.canta()*, e il metodo *CantanteLirico.produciAcuto()* è implementato in *Pavarotti.produciAcuto()*. Quindi il compilatore non si troverà mai nell'imbarazzante situazione di non sapere dove andare a pescare il codice di un metodo. Come molti altri aspetti della programmazione a oggetti, le classi astratte sono uno strumento potente ma che ti potrebbe lasciare perplesso fino a quando non ti sarà diventato familiare. L'idea della classe astratta è che sto separando l'interfaccia di una classe (i nomi e i parametri dei metodi pubblici) dalla sua implementazione (il codice dei metodi). A volte interfaccia e implementazione vanno in coppia, ma in molti casi mi interessa solo la prima. Ad esempio, il metodo polimorfico *Concerto.iniziaConcerto()* non è interessato a cosa succede nel metodo *canta()*. Tutto quello che gli interessa è che questo metodo esiste. Quindi il *Concerto* parla all'interfaccia della classe *Cantante,* ma si disinteressa dell'implementazione.

SEMPRE PIÙ IN ALTO

Possiamo fare ancora un altro passo in questo nostro viaggio verso classi sempre più astratte. Se torni a guardare le classi *Cantante* e *CantanteLirico* scoprirai che appartengono a due categorie diverse. *CantanteLirico* contiene alcuni metodi astratti, ma anche un po' di codice. *Cantante,* invece, ha solo un metodo astratto e non contiene nemmeno una riga di codice concreto. È una classe puramente astratta, una semplice interfaccia senza ombra di implementazione. In casi come questo ci conviene usare la parola chiave *interface* al posto di *class.* Una *interface* è come una *class,* ma il compilatore esige che tutti i suoi metodi siano astratti e pubblici. Infatti non sei nemmeno obbligato ad usare le parole chiave *abstract* e public per i metodi di un'interfaccia: le due cose sono sempre sottintese (anche se io preferisco specificare comunque il *public*):

```
public interface Cantante {public void canta();}
```

EREDITARIETÀ OBBLIGATORIA

Un metodo astratto non ha un'implementazione (cioè non contiene codice), quindi il compilatore non saprebbe come costruirlo. Questo è il motivo per cui se una classe ha anche un solo metodo astratto, allora la classe stessa deve essere astratta. D'altra parte, una classe astratta può anche non avere alcun metodo astratto. In questo caso il comportamento della classe è completamente definito, ma ciononostante non puoi chiamare il costruttore della classe.

Dichiara *abstract* la classe *CantanteLirico.* Secondo te, questa classe contiene qualche metodo che può diventare astratto? Se sì, dichiara abstract anche questi metodi.

Una *interface* è quanto di più "ideale" esista. Non contiene codice eseguibile. Si limita a dire: ecco un nuovo tipo di oggetti, ed ecco attraverso quali metodi puoi comunicare con loro. Se voglio fornire un'implementazione all'interfaccia devo scrivere una classe che "implementa" l'interfaccia. Per implementare un'interfaccia non devo usare la parola *extends,* ma una nuova parola chiave: *implements.*

```
public class Eminem implements Cantante {<...>}
```

Ora *Cantante* è un'interfaccia, quindi non deve più essere "estesa", ma implementata. Devo applicare questa modifica a tutte le sottoclassi, compresa *CantanteLirico.* Per essere precisi non possiamo nemmeno più parlare di "sottoclassi", ma al limite di sottotipi. Nota anche che il programma *Concerto* non cambia: che si tratti di classi astratte o di interfacce, per lui sono semplicemente tipi di oggetti. Quindi può continuare serenamente a creare un oggetto concreto e a fare poi un upcasting ad un suo "supertipo", classe o interfaccia che sia.

PERCHÉ TUTTO QUESTO?

Questo discorso sulle interfacce può lasciarti perplesso. Non sarebbe meglio usare semplicemente le classi astratte ed evitare queste nuove parole chiave? Per capire perché esistono le interfacce ti devo parlare di una cosa che in Java non esiste: l'ereditarietà multipla. In alcuni linguaggi, come il C++, una classe può ereditare da più classi contemporaneamente. In Java questo non è possibile:

```
class Idrovolante extends Aeroplano, Barca { // non
                                    compila in Java <...> }
```

Per quale motivo Java considera illegale una funzionalità così utile? Perché i creatori di Java volevano un linguaggio semplice, e l'ereditarietà multipla può creare ogni sorta di problemi molto complicati. Cosa succede, ad esempio, se sia la classe *Aeroplano* che la classe *Barca* definiscono un metodo *accelera()*? Quale codice viene eseguito se chiamo *accelera()* su un *Idrovolante?* Peggio ancora, cosa succede se sia *Aeroplano* che *Barca* derivano da una classe *Veicolo,* che definisce un campo velocità? Il campo viene ereditato da entrambe le classi, e quando arriva all'Idrovolante è diventato in effetti due campi separati! Come puoi immaginare, questo genere di problemi può diventare un incubo nei linguaggi che consentono l'ereditarietà multipla. Questi problemi hanno una radice comune: più campi o più metodi con la stessa interfaccia, ma implementazioni diverse. Se ho sempre una sola implementazione, non ci sono dubbi su cosa succede quando chiamo *accelera()*: anche se l'*Aeroplano* e la *Barca* hanno entrambi questo metodo nelle loro interfacce, il sistema potrà sempre chiamare l'unica implementazione disponibile. Per questo motivo una classe Java può implementare tutte le interfacce che vuole, ma può ereditare da una sola classe:

```
class Idrovolante extends Aeroplano implements Natante,
                                    Veicolo, Trasporto {
  <...>}
```

Ecco a cosa servono le interfacce: a rimuovere il vincolo che ci obbliga a definire un tipo in base ad un unico altro tipo. Posso definire un'interfaccia *Musicista,* e creare una classe che è allo stesso tempo un *Cantante* e un *Musicista:*

```
public interface Musicista {
  public void suona();}
public class Sting implements Cantante, Musicista {
  public void canta() {
    System.out.println("Message in a bottle...");}
  public void suona() {
    System.out.println("plin, plon..."); } }
```

Naturalmente *Sting* deve implementare per conto proprio tutti i metodi definiti dai suoi supertipi, tranne quelli che hanno già un'implementazione. A questo punto un oggetto di classe *Sting* appartiene a più tipi diversi, e grazie al polimorfismo lo posso usare in situazioni completamente differenti:

Fig. 2: *Diagramma delle classi in UML 2*

```
public class MusicaEParole {
  public static void main(String[] args) {
    Sting sting = new Sting();
    invitaCantanteSulPalco(sting);
    invitaMusicistaSulPalco(sting);}
  private static void invitaCantanteSulPalco(Cantante c) {
    c.canta();}
  private static void invitaMusicistaSulPalco(Musicista m) {
    m.suona();}
}
```

Il risultato di questo programma è:

Message in a bottle...
plin, plon...

Eccoti una versione aggiornata del nostro diagramma delle classi. Nella nuova notazione UML 2 si può usare un cerchietto per rappresentare le interfacce.

Paolo Perrotta

ESERCIZIO 3

Scrivi una sottoclasse concreta di *CantanteLirico* che non ridefinisce il metodo *produciAcuto()*. Cosa dice il compilatore? Prova a dichiarare abstract questa sottoclasse. Cosa succede ora?

NOTA

TIPI E CLASSI

Nelle prime lezioni di questo corso ti ho detto che le parole "tipo" e "classe" in Java sono praticamente intercambiabili, con l'eccezione dei tipi primitivi come int e char. In realtà ho detto una piccola bugia: in Java esiste un concetto di "tipo" molto più astratto, rappresentato dalle interfacce. Una interface definisce un nuovo tipo esattamente come una class, ma non vincola questo tipo a nessuna particolare implementazione.

Funzioni di base: anteprima e stampa

L'oggetto Printer e i DB

Descriveremo come implementare un *Viewports*, e illustreremo alcune nuove caratteristiche dell'oggetto *Printer*. Sarà l'occasione per integrare il progetto introdotto nel precedente appuntamento

Fig. 1: ***Il form che contiene il Viewports***

Nel precedente appuntamento abbiamo descritto gli strumenti che si possono utilizzare per realizzare l'anteprima e la stampa di report: dagli oggetti *Printer* e *Picture Box,* all'oggetto *DataReport.* Come avete potuto accertare, l'oggetto *Printer* è utile (essendo un controllo nativo) ma molto difficile da gestire, soprattutto quando non è pienamente compatibile con i driver della vostra stampante. Il *DataReport,* invece, è semplice da utilizzare e facilmente collegabile ad un database ma, come ogni controllo, ha i suoi svantaggi in termini di portabilità, flessibilità e di compatibilità con i vecchi progetti. In questo appuntamento, dunque, completeremo l'applicazione introdotta nel precedente articolo, descrivendo una finestra, tipo *Viewports,* che consente di "esplorare" la *PictureBox* che mostra l'anteprima del report di stampa. Inoltre, superando non poche difficoltà, vedremo come controllare, sempre attraverso *Printer* e *PictureBox* un report di più pagina collegato al database *Biblio.mdb* (fornito con Visual Basic). Iniziamo presentando il *Viewports.*

IL VIEWPORTS

Il nostro Viewports si basa su due PictureBox, *Picture1* e *Picture2,* e due scrollbar, *HScroll1* e *VScroll1.* Questi ultimi servono per controllare la *Picture2* (che contiene l'immagine o il report da esplorare) dentro la *Picture1.* Create, dunque, un nuovo progetto (o inserite un form nel progetto del precedente appuntamento) e nel *form1* disponete gli elementi del Viewports come mostrato in Fig. 1. Inoltre, per i comandi, inserite un menu a tendina con due voci: *caricaimmagini* e *Viewports.* Il primo comando serve per caricare un'immagine in *Picture2,* il secondo, invece, attiva il *Viewports.* Descriviamo il codice partendo dai menu. Il codice per caricare l'immagine è banale e non lo descriviamo. Per quanto riguarda il menu *Viewports*, dovete predisporre il codice:

```
Private Sub mnuViewports_Click()
 ScaleMode = vbCentimeters
 Picture1.ScaleMode = vbCentimeters
 Picture2.AutoSize = True
 Form_Resize
End Sub
```

Nella *mnuViewports_Click* dopo aver impostato la scala, per il *Form* e la *Picture1,* si pone *Picture2* in *autosize* e si avvia la *FormResize.*

```
Private Sub Form_Resize()
 Picture1.Height = Me.Height
 Picture1.Width = Me.Width
 Picture1.Move 0, 0, ScaleWidth - VScroll1.Width, _
 ScaleHeight - HScroll1.Height
 Picture2.Move 0, 0
 HScroll1.Top = Picture1.Height
 HScroll1.Left = 0
 HScroll1.Width = Picture1.Width
 VScroll1.Top = 0
 VScroll1.Left = Picture1.Width
 VScroll1.Height = Picture1.Height
 HScroll1.Max = Picture2.Width - Picture1.Width
 VScroll1.Max = Picture2.Height - Picture1.Height
 VScroll1.Visible = (Picture1.Height < Picture2.Height)
 HScroll1.Visible = (Picture1.Width < Picture2.Width)
End Sub
```

Nella *Resize* si fa in modo che la *Picture1* occupi tutto lo spazio del Form meno le dimensioni degli *Scroll.* Poi si riposizionano gli *Scroll* e si stabilisce, in base alle dimensioni delle *Picture,* quando devono essere visibili. Per gestire il movimento della *Picture2,* invece, bisogna prevedere il seguente codice.

```
Private Sub VScroll1_Change()
  Picture2.Top = -VScroll1.Value
End Sub
Private Sub HScroll1_Change()
   Picture2.Left = -HScroll1.Value
End Sub
```

Con *VScroll* controlliamo la proprietà *Top* e con *HScroll* la proprietà *Left.* Notate che assegniamo il *Value* negativo così si ottiene un movimento opposto al senso in cui si preme il pulsante dello *Scroll.* Adesso descriviamo come gestire dei report di carat-

teri. In particolare, prima introduciamo un report di una sola pagina riempito con un carattere, poi (dopo aver integrato il progetto con il Viewports) vediamo come gestire un report di più pagine.

UNA PAGINA PIENA DI CARATTERI

La procedura seguente, nominata *Caratteritest*, permette di riempire una pagina con la vocale *"a"* (in minuscolo) e di circondare il tutto con i caratteri "|" (sui due lati) e *"_"* (prima e ultima riga). La *Caratteritest* ha due argomenti: *antestampa* che può essere l'oggetto *Printer* o la *Picture3* (come capiremo tra poco) e *Prop* che è il rapporto di "proporzionalità" calcolato attraverso la procedura *Impostaoggetti* (descritta nel precedente appuntamento e che più avanti modificheremo). Se volete usare *Caratteritest*, subito, senza aspettare l'integrazione dell'applicazione, sostituite antestampa con *Printer* ed inserite il tutto in un *CommandButton*. Infatti, come saprete, il parametro *Prop* non è necessario, quando si usa *Printer*.

```
Private Sub Caratteritest(antestampa As Object, _
                         Optional Prop As Double = 1)
...
End Sub
```

Nella *Caratteritest* impostiamo innanzi tutto l'unità di misura ed il *Font*. La scelta del *Font* è importantissima: come capirete testando la vostra applicazione, il tipo di *Font* condiziona il funzionamento dell'applicazione, soprattutto dell'anteprima. Per questo, nei nostri test abbiamo preferito lo *"Smart Font"*.
Prima di procedere, puntualizziamo un aspetto importante che riguarda la scelta della stampante, per i nostri test abbiamo utilizzato *"Microsoft Office Document Image Writer"* che consente di testare le funzionalità dell'oggetto *Printer* anche se non si ha una stampante in linea. Ora descriviamo come inserire il *Viewports* e la *Caratteritest* nell'applicazione che abbiamo introdotto nel precedente appuntamento.

INTEGRIAMO IL PROGETTO

Come si può constatare dalla Fig. 2, l'interfaccia dell'applicazione è stata completamente ridisegnata. Per questo, conviene elencare nuovamente gli elementi del form. In particolare bisogna inserire:

1. gli oggetti che compongono il *Viewports*, tranne il menu a tendina;
2. due *PictureBox* e un *CommanDialog*, una *PictureBox (Picture2)* sarà il contenitore di tutti gli elementi che non appartengono al *Viewports* l'altra, invece, conterà l'immagine di sfondo;
3. un frame e quattro pulsanti (*Preview, Stampa, Stampanti* e *Annulla anteprima*);
4. un secondo frame con dentro un pulsante con un'immagine (una lente d'ingrandimento) e un checkbox, questi serviranno per controllare il resize dell'anteprima;
5. un terzo frame con un controllo *SpinButton* e due textbox che utilizzeremo per controllare l'anteprima di report di più pagine.

Fig. 2: ***L'applicazione che gestisce i report di stampa***

Secondo quanto elencato, *Picture1* e *Picture2* sono gli unici elementi in contatto diretto con il form (oggetto *Container*) questo ci tornerà utile per il resize. Nella parte dichiarativa aggiungiamo le seguenti:

```
Dim SaltaPreview   As Boolean
Dim SceltaPreview As Boolean
```

Le due variabili sono utilizzate, rispettivamente, per evitare l'esecuzione della *Preview_Click* e per segnalare che è stato premuto il pulsante *Preview*, come sarà chiaro tra poco. Facciamo notare che all'avvio il *Framezoom* (con lente d'ingrandimento e *CheckScroll*) è disattivo, esso viene attivato quando si preme il pulsante *Preview;* inoltre facciamo in modo che le barre di scorrimento compaiono solo se si seleziona il relativo *CheckBox* e si preme il pulsante con la lente, mentre si disattivano quando il *CheckBox* non è selezionato. Iniziamo la carrellata di codice cominciando dalla *Form_Load*.

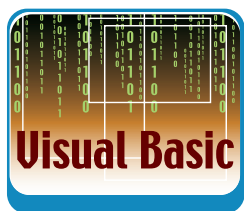
```
Private Sub Form_Load()
 CommonDialog1.CancelError = True
 SaltaPreview = True
 Me.HScroll1.LargeChange = 100
 Me.HScroll1.SmallChange = 100
 Me.VScroll1.LargeChange = 100
 Me.VScroll1.SmallChange = 100
End Sub
```

Nella *Form_Load* abbiamo previsto le istruzioni che permettono d'impostare lo spostamento degli *Scroll*.

IL RESIZE DEL FORM

Nella *Form_Resize*, invece, riposizioniamo il *ViewPorts* lasciando invariato le dimensioni della *Pic-*

VSCROLL E HSCROLL

Le Scrollbar forniscono un meccanismo per controllare le proprietà che dipendono da valori numerici; per esempio la posizione (*Top* e *Left*) di un controllo su una form. Le caratteristiche principali dei controlli *VscrollBar* e *HscrollBar* sono: *Value*, ritorna o setta il valore corrispondente alla posizione del controllo; *Max* e *Min*, ritornano o settano i valori massimo e minimo della proprietà *value*; *LargeChange:* setta o ritorna la dimensione dello spostamento quando si clicca sull'area tra la freccia e lo scroll box; *SmallChange*, setta lo spostamento fatto quando si clicca sulla freccia della Scrollbar.

ture2 (*PictureBox* che contiene i comandi).

```
Private Sub Form_Resize()
 On Error Resume Next
 Picture1.Height = Me.Height
 Picture1.Width = Me.Width - Me.Picture2.Width
 Picture1.Move 0, 0, ScaleWidth - VScroll1.Width _
   - Me.Picture2.Width, ScaleHeight - HScroll1.Height
 Picture2.Move Picture1.Width + VScroll1.Width, 0
 If SaltaPreview = False Then
  Preview_Click
 End If
...
End Sub
```

Notate che nella *Resize,* se *SaltaPreview* assume il valore *False* viene eseguita la *Preview_Click*, questo succede quando si clicca il pulsante con la lente o il *CheckScroll.* Ora vediamo come bisogna cambiare la *Preview_ Click* per includere l'ingrandimento e la stampa di più pagine.

NOTA

MICROSOFT OFFICE DOCUMENT

Microsoft Office Document Image Writer **(stampante) e *Microsoft Office Document Imaging* consentono di gestire i documenti "digitalizzati" come qualsiasi altro documento di Microsoft Office. Si rivelano utili quando non si ha una stampante in linea e si vogliono testare le funzionalità di stampa di un'applicazione (oppure per controllare l'anteprima dei fax presenti nelle cartelle della Console Servizio Fax, fare il riconoscimento ottico dei caratteri da immagini digitalizzate, ecc). Essi usano il formato Tiff e Mdi.**

IL PREVIEW

La procedura per il *Preview* è la seguente

```
Private Sub Preview_Click()
 Dim Proporz As Double
 SaltaPreview = True
 SceltaPreview = True
 Framezoom.Enabled = True
 Picture3.Cls
 AmpiezzaPag = Printer.Width / 567
 AltezzaPag = Printer.Height / 567
 If Checkscroll.Value = 0 Then
  Proporz = ImpostaOggetto(Picture3)
  AnteprimaStampa Picture3, Proporz
 Else
  Picture3.Move 0, 0, Printer.Width, Printer.Height
  Proporz = ImpostaOggetto(Picture3)
  AnteprimaStampa Picture3
 End If
End Sub
```

Nella *Preview_Click* in base a *CheckScroll.Value* si decide come eseguire la *ImpostaOggetto* e la *AnteprimaStampa.*
La funzione *ImpostaOggetto,* che calcola il rapporto di proporzione e la scala per la *Picture3,* è presentata sotto. In essa abbiamo introdotto le istruzioni che permettono di impostare le dimensioni della *Picture3* in base al *Checkscroll.Value.*

```
Private Function ImpostaOggetto _
                        (OggAnteprima As Object) As Double
...
 End Function
```

Qualche cambiamento dobbiamo farlo anche alla *Stampa_Click.*

```
Private Sub Stampa_Click()
On Error GoTo errore
 SceltaPreview = False
 Printer.ScaleMode = vbCentimeters
 AnteprimaStampa Printer
 Printer.EndDoc
 Exit Sub
errore:
 MsgBox ("Errore su Stampante")
 Err.Clear
End Sub
```

In particolare impostiamo il valore di *SceltaPreview,* necessario per la procedura che permette di fare l'anteprima o stampare più pagine. La procedura *AnteprimaStampa,* infine, richiama la procedura *Caratteritest* introdotta in precedenza.

```
Private Sub AnteprimaStampa(OggAnteprima As
                        Object, Optional Prop As Double = 1)
  Caratteritest OggAnteprima, Prop
End Sub
```

Dalla procedura *AnteprimaStampa* potete anche avviare il codice introdotto nel precedente appuntamento (con il quale abbiamo ricavato il form di Fig. 2) e la procedura che permette di gestire report di più pagine. Quest'ultima l'abbiamo nominata *Piupaginetest* e può essere richiamata attraverso la seguente riga di codice.

```
 Piupaginetest OggAnteprima, Prop
```

Naturalmente, per gestire queste funzionalità, bisogna predisporre altri pulsanti sul Form ed un'istruzione *Case* nella *AnteprimaStampa.* Prima d'illustrare i report di più pagine descriviamo le procedure per gestire lo *Zoom* e lo *Scroll.*

ZOOM E SCROLL

Nel pulsante zoom prevediamo il seguente codice:

```
Private Sub zoom_Click()
 If WindowState = 0 Then
  Picture3.Cls
  SaltaPreview = False
  Me.WindowState = 2
 Else
  Picture3.Cls
  SaltaPreview = False
  Me.WindowState = 0
 End If
End Sub
```

Anche in questo caso usiamo il *SaltaPreview* per eseguire la *Preview_Click* nella *Resize*. Nella *Checkscroll_Click*, mostrata sotto, invece il *SaltaPreview* verrà impostato a *False* solo quando il *Ckeckscroll* non è selezionato e le barre di scorrimento sono visibili.

```
Private Sub Checkscroll_Click()
 Preview.Enabled = Not Preview.Enabled
 If Me.Checkscroll.Value = 0 And (VScroll1.Visible =
                        True Or HScroll1.Visible = True) Then
  SaltaPreview = False
  Form_Resize
 End If
End Sub
```

Sul *Form* abbiamo previsto anche il pulsante per annullare l'anteprima:

```
Private Sub annulla_Click()
 SceltaPreview = False
 Picture3.Cls
 SpinButton1.Enabled = False
 Framezoom = False
End Sub
```

Facciamo notare il controllo *SpinButton1*, che introdurremo tra poco, è attivo solo quando si gestiscono report di più pagine.

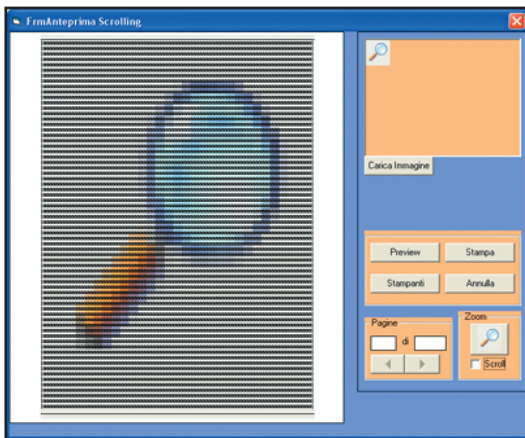



Fig. 3: ***L'anteprima di una pagina riempita di caratteri***

REPORT DI PIÙ PAGINE

Nel CD allegato alla rivista troverete il progetto presentato in questo e nel precedente appuntamento e le procedure *Piupaginetest* e caricapagina (che per ragione di spazio non abbiamo potuto illustrare). Queste procedure permettono di creare un *Report* con i dati della tabella *Authors* del database *Biblio .mdb*. La *Piupaginetest*, sulla base delle dimensioni del foglio della stampante e del Font, calcola quanti caratteri possono essere inseriti in una riga e di quante righe è formata una pagina. Quest'ultimo valore viene utilizzato per stabilire la dimensione della pagina del *RecordSet* collegato alla Tabella *Authors (RstAutore.PageSize)*, ecc. Dunque si fa corrispondere la dimensione della pagina di stampa (in righe) con quella della pagina del *RecordSet* (espressa in record), così diventa più semplice scorrere le pagine nell'anteprima. Le pagine da stampare o mostrare in anteprima vengono caricate attraverso la caricapagina.

Di seguito riportiamo l'interfaccia delle due procedure e il codice per lo *SpinButton1* che permette di scorrere le pagine.

```
Private Sub Piupaginetest(antestampa As Object, _
 Optional Prop As Double = 1)
 ` nel CD allegato alla rivista
End Sub
Sub caricapagina(antestampa As Object, _
 Optional numero As Integer = 0)
 ` nel CD allegato
End Sub
```

Per lo *SpinButton1*, invece, prevediamo la *SpinDown*, per lo spostamento indietro, e *SpinUp* per lo spostamento in avanti.

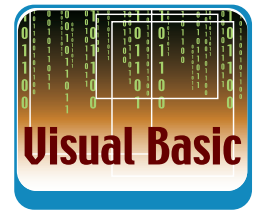
```
Private Sub SpinButton1_SpinDown()
 If CInt(txtnpagine) > 1 Then
  txtnpagine = txtnpagine - 1
  Picture3.Cls
  caricapagina Picture3, txtnpagine
 End If
End Sub
Private Sub SpinButton1_SpinUp()
 If CInt(txtnpagine) < CInt(txtmaxpag) Then
  txtnpagine = txtnpagine + 1
  Picture3.Cls
  caricapagina Picture3, txtnpagine
 End If
End Sub
```

Notate che *Txtpagina* contiene la pagina corrente (sul form è presente anche *txtmaxpag* che contiene il numero totale di pagine).

CONCLUSIONI

In questo e nel precedente appuntamento abbiamo descritto vari metodi per realizzare l'anteprima e la stampa di report. Ora resta da integrare le procedure che trovate nel CD allegato alla rivista.

Vi preannunciamo che il compito non è banale, infatti, basta un Font sbagliato o un driver di stampante non compatibile per tenervi occupati per diverse ore!

Massimo Autiero

PROPRIETÀ RECORDSET

Per un *RecordSet* una pagina è un gruppo di record controllabili attraverso le seguenti proprietà:

- ***PageCount*, che restituisce il numero di pagine di dati contenute nel *RecordSet*;**
- ***PageSize*, che restituisce o imposta un long che indica il numero di record che formano una pagina logica di dati;**
- ***AbsolutePage*, che imposta o restituisce la pagina che contiene il record corrente.**

Naturalmente l'ultima pagina, di solito, ha un numero di record inferiore a *PageSize*. Queste proprietà le abbiamo utilizzate per creare il report di stampa collegato alla tabella *Authors* di *Biblio.mdb*.

Serializzazione/deserializzazione di classi da e verso XML

.NET XML Serializer

Le classi sono popolate a partire dai dati presenti in un database con un processo manuale a carico dello sviluppatore ma con .NET la serializzazione/deserializzazione di classi avviene in modo automatico

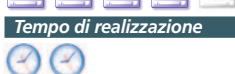
Le vostre classi *Fattura, Cliente* e *Indirizzo,* dotate delle immancabili proprietà *NumeroFattura, PartitaIVA, CAP* e numerose altre che ometto per evitare di annoiarvi, hanno tutte una caratteristica comune: il loro popolamento a partire da una fonte dati e la persistenza delle informazioni verso una fonte dati è sempre un'operazione manuale. Centinaia di linee di codice che, dopo l'immancabile invocazione di query o di stored procedure nel database, procedono prima alla creazione delle istanze e poi alla valorizzazione delle proprietà delle classi, con un analogo processo all'inverso per quanto riguarda la persistenza nel database delle informazioni aggiunte o modificate nelle classi. Grazie a .NET è disponibile la tecnologia XML *Serializer:* se si possiede uno schema XML, un tool *(xsd.exe)* del framework consente di generare il sorgente di una classe in uno dei linguaggi del framework. A questo punto la tecnologia è in grado creare a runtime delle istanze di classi nella struttura descritta dallo schema XML, a partire da un file XML che risponde allo schema in questione e, viceversa, di persistere su file XML il contenuto di istanze classi prodotte con lo stesso principio.

XML SERIALIZER

Esiste infatti un potente oggetto del framework, definito nel namespace *System.Xml.Serialization,* e cioè *XmlSerializer.* Esso agisce su oggetti che devono soddisfare certi requisiti ed è in grado di serializzare soltanto le proprietà e i campi pubblici dell'oggetto e non può nulla, invece, su *indexer* e *metodi.* L'altro requisito richiesto alle classi da serializzare è la presenza del costruttore di default:

```
'Visual Basic
Public Sub New()
   'dummy
End Sub
//C#
public nomeClasse() {
   //dummy
}
```

Si supponga di voler serializzare/deserializzare una rubrica telefonica costituita da voci definite nella seguente classe:

```
`Visual Basic
   Public Class PhoneBookEntry
      Public Sub New()
         'dummy
      End Sub
      Public Position As Integer
      Public PhoneBookName As String
      Public Name As String
      Public Number As String
      Public EntryType As PhoneBookEntryType
      Public ReadOnlyEntry As Boolean
   End Class
   Public Enum PhoneBookEntryType
      ScPETNormal
      ScPETInternational
   End Enum
//C#
   public class PhoneBookEntry {
      PhoneBookEntry () { //dummy }
      public int Position;
      public string PhoneBookName;
      public string Name;
      public string Number;
      public PhoneBookEntryType EntryType;
      public bool ReadOnlyEntry; }
   public enum PhoneBookEntryType {
      ScPETNormal,
      ScPETInternational  }
```

Possiamo osservare la semplicità della classe e, soprattutto, il fatto di non aver adoperato nessuna esotica forma di definizione delle informazioni. Tutte le voci di rubrica sono contenuti in un classe *SimpleSerializablePhonebook*:

```
'Visual Basic
   <XmlRootAttribute("GSMProgramming",
                              IsNullable:=False)> _
   Public Class SimpleSerializablePhonebook
   …
      <XmlAttributeAttribute("ReadOnly")> Public
                     ReadOnlyPhonebook As Boolean
   End Class
//C#
      [XmlRootAttribute("GSMProgramming",
                              IsNullable=False)]
      public class SimpleSerializablePhonebook {
         SimpleSerializablePhonebook () {
         //dummy}
      [XmlArray("PhoneBook"), XmlArrayItem("Entry",
                              IsNullable=True)]
      …
      [XmlAttributeAttribute("Count")] public int Count;
      [XmlAttributeAttribute("ReadOnly")] public bool
                              ReadOnlyPhonebook;}
```

Fig. 1: **Il file xml della rubrica**

Essa presenta le proprietà tipiche di una collezione di dati quali *Name, StartIndex, EndIndex, Count, Capacity.* Inoltre, presenta la proprietà *Entries()* che altro non è che un vettore di oggetti *PhonebookEntry* che contiene appunto l'elenco di voci associate alla rubrica. Fornisce inoltre un costruttore di default come richiesto dall'oggetto di serializzazione. Si noterà inoltre l'attributo *XmlRootAttribute* definito a livello di classe: esso fa parte del set di attributo di *XMLSerializer* ed impone al serializzatore, in fase di creazione del file XML, di creare un oggetto radice con il nome di *"GsmProgramming".* Anche a livello dei singoli campi della classe sono assegnati degli attributi, si tratta di *XmlAttributeAttribute* che impone il nome che, il campo su cui insiste l'attributo, in fase di serializzazione in XML verrà definito come un attributo del nodo XML definito per quella classe. Infine, per *Entries,* è definito anche un altro attributo, *XmlArrayItem,* che istruisce il serializzatore sul quale assegnare ai nodi XML che verrano creati per ciascun elemento dell'array, in pratica per ciascuna voce di rubrica. Ma osserviamo il semplice codice che è in grado di persistere su un file XML un'intera rubrica:

```
'Visual Basic
Public Function ExportPhoneBookToFile(ByVal fileName
   As String, phonebook As SimpleSerializablePhonebook)
                                          As Boolean
   Try
      Dim l_oXmlSer As New XmlSerializer(GetType(
```

```
                        SimpleSerializablePhonebook))
      Dim l_oFs As New FileStream(fileName,
                                    FileMode.Create)
      l_oXmlSer.Serialize(l_oFs, phonebook)
      l_oFs.Close()
      Return True
   Catch
      Return False
   End Try
End Function
//C#
public bool ExportPhoneBookToFile(string fileName As
   String, SimpleSerializablePhonebook phonebook) {
   try {
      XmlSerializer l_oXmlSer = New XmlSerializer(
            typeof(SimpleSerializablePhonebook));
      FileStream l_oFs = New FileStream(fileName,
                                    FileMode.Create);
      l_oXmlSer.Serialize(l_oFs, phonebook);
      l_oFs.Close();
      return true;}
   catch {
      return false;}}
```

Alla funzione vengono passati semplicemente un'istanza dell'oggetto *Phonebook* e il nome del file su cui effettuare il salvataggio. La funzione crea un'istanza di *SimpleSerializablePhonebook* a partire dall'oggetto *Phonebook* passato. A questo punto viene istanziato il serializzatore e un oggetto *FileStream,* in grado si salvare stream su file, e il serializzatore effettua dunque la serializzazione sul file XML del contenuto del *phonebook.* In Fig. 1 possiamo osservare il file XML prodotto, visualizzato direttamente da Internet Explorer che è in grado di leggerli e di formattarli a video. Potrete notare la presenza del nodo principale *GSMProgramming* **[2]** nel quale sono definiti gli attributi *Name, StartIndex, EndIndex, Count, Capacity* e *ReadOnly,* come descritto dagli attributi relativi che abbiamo esaminato in precedenza. Ciascuna voce di rubrica invece è inserita in un nodo di tipo *Entry.* Per effettuare l'operazione contraria, cioè per leggere un phonebook salvato su file XML, definiamo la seguente funzione:

```
'Visual Basic
Public Function ImportPhonebookFromFile(ByVal
   fileName As String) As SimpleSerializablePhonebook
   Try
      Dim l_oXmlSer As New XmlSerializer(GetType(
                        SimpleSerializablePhonebook))
      Dim l_oFs As New FileStream(fileName,
                                    FileMode.Open)
      Dim l_oSP As SimpleSerializablePhonebook = _
            CType(l_oXmlSer.Deserialize(l_oFs),
                        SimpleSerializablePhonebook)
      Return l_oSP
   Catch
```

```
        Return Nothing
    End Try
End Function
'C#
public SimpleSerializablePhonebook
              ImportPhonebookFromFile(string fileName) {
    try
        XmlSerializer l_oXmlSer = new XmlSerializer(
                     typeof(SimpleSerializablePhonebook));
        FileStream l_oFs = new FileStream(fileName,
                                         FileMode.Open);
        SimpleSerializablePhonebook l_oSP = (Simple
        SerializablePhonebook)l_oXmlSer.Deserialize(l_oFs);
        return l_oSP; }
    catch {return null; } }
```

Essa accetta come parametro il nome del file XML e restituisce un oggetto *SimpleSerializablePhonebook*. Effettua dunque l'operazione all'inverso: istanzia un serializzatore, istanzia un *FileStream* col quale apre il file XML e, col metodo *Deserialize()* produce un'istanza del solito *SimpleSerializablePhonebook* a partire dal file XML. Infine istanzia un oggetto *SimpleSerializablePhonebook*.

UN PROCESSO AUTOMATICO: XSD.EXE

Sebbene il meccanismo di serializzazione e di deserializzazione sia del tutto automatico, resta la necessita di scrivere la classe di mappatura dell'xml e di farlo secondo le regole di compatibilità prescritte dalla tecnologia. Così gli ingegneri di Microsoft hanno pensato di semplificare l'esistenza degli sviluppatori fino in fondo, fornendo un semplice tool in grado di generare le classi sorgente a partire dall'xml di origine. L'utility in questione è *xsd.exe* che è disponibile nella sottodirectory *\SDK\v1.1\Bin* del path di installazione di Visual Studio .NET 2003 (ma esiste l'omologo in Visual Studio .NET 2002). In realtà il processo non è così diretto, ma è un po' più articolato perché si basa sullo schema del file XML di origine per generare le classi. Senza voler ripetere cose note ai più e rimandando a **[3]** per coloro che vogliano approfondire l'argomento, uno schema xml è sostanzialmente un file XML che, seguendo alcune convenzioni, definisce la struttura semantica di una specifica categoria di file XML. Infatti il formato XML in se già prevede delle regole di validazione sintattica che rendono facilmente valutabile la bontà del file e cioè il suo essere well formed o meno. L'XML in se, però, non è in grado di determinare la semantica del file e cioè, ad esempio, impone che ci sia un solo nodo *root,* ma non prescrive che si tratti di un nodo *Order,* anziché *Invoice* e così via. Ed è giusto che non lo faccia proprio perché l'XML è una notazione universale ed utilizzabile per rappresentare set di informazioni diversissimi tra loro. Gli schemi, invece, descrivono proprio le regole mancanti nativamente all'XML.

Consideriamo il seguente file XML *FileSchema.xml* di esempio:

```
<?xml version="1.0" encoding="utf-8" ?>
<InputDefinition>
<VerifyRecordNumber>true</VerifyRecordNumber>
  <File>
    <Name>CENTRE.exp</Name>
    ...
     </Field>
     <Field>
       <Name>PT_PatientID</Name>
       <ExportName>PT_LASTNAME</ExportName>
     </Field>
  </File>
</InputDefinition>
```

La struttura logica del file può essere in qualche modo dedotta dalla semplice osservazione. Infatti possiamo avventurarci in alcune considerazioni quali:

- Il nodo root deve essere di tipo *InputDefinition;*
- Deve essere presente un sottonodo *VerifyNumber;*
- Possono essere presenti uno o più sottonodi complessi *Field;*
- Ciascun sottonodo *Field* è composto da due altri sottonodi *Name* ed *ExportName.*

Potremmo scrivere a mano il file schema che descrive le regole di questo schema, ma l'utility xsd può farlo per noi eseguendoli con i seguenti parametri nella directory dove è presente il file xml di origine:

```
xsd.exe fileschema.xml
```

Esso, innanzitutto verificherà che alcune regole siano rispettate e, in caso affermativo, genererà il file omonimo a quello XML di origine, ma con estensione *.xsd (Xml Schema Definition):*

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema id="NewDataSet" xmlns=""
    xmlns:xs = "http://www.w3.org/2001/XMLSchema"
  xmlns:msdata="urn:schemas-microsoft-com:
                                        xml-msdata">
 ...
    <xs:complexType>
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="InputDefinition" />
    </xs:choice>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

[1] Sezione MSDN relativo a .NET - http://msdn.microsoft.com/dotnet

[2] PROGRAMMARE IL CELLULARE ***Vito Vessia*** **Hoepli 2002**

[3] Definizione degli xml schema http://www.w3.org/XML/Schema

In Fig. 2 è possibile osservare la rappresentazione grafica che fa Visual Studio .NET 2003 dello schema in questione. In realtà l'utility non è in grado di generare lo schema, ma semplicemente prova a dedurlo, infatti è possibile sempre far discendere un caso specifico da una regola generale, ma l'operazione inversa è decisamente più empirica e quindi non può che considerarsi solo un tentativo. Infatti *xsd.exe* NON può sapere che il nodo *VerifyRecordNumber* deve essere necessariamente presente e non più di una volta, ma si limita a prescrivere che debba essere presente almeno una volta *(minOccurs="0")* senza prescrivere nulla sul numero massimo di occorrenze. L'ideale sarebbe, infatti, procedere con un raffinamento manuale dello schema a partire da quello generato in modo automatico. Una volta ottenuto lo schema si può procedere con la generazione del sorgente vero e proprio utilizzando i seguenti parametri di xsd.exe:

```
xsd.exe /classes /language:CS fileschema.xsd
```

AMMIRIAMO IL RISULTATO

Il parametro più importante è il file schema di origine a cui si deve aggiungere il linguaggio del sorgente da generare con il parametro */language* (*CS* per C# e VB per Visual Basic .NET) e con il parametro */classes* si indica la generazione di una normale classe flat come quelle scritte nell'esempio precedente del *Phonebook*. Ed ecco finalmente il risultato del processo di generazione:

```
//--------------------------------------------------------
// <autogenerated>
//     This code was generated by a tool.
//     Runtime Version: 1.1.4322.573
//     Changes to this file may cause incorrect behavior
                                        and will be lost if
...
    public string ExportName;
}
    [System.Xml.Serialization.XmlRootAttribute(
                        Namespace="", IsNullable=false)]
public class NewDataSet {
    [System.Xml.Serialization.XmlElementAttribute(
                                        "InputDefinition")]
    public InputDefinition[] Items;
}
```

Si osserverà come siano presenti gli attributi già descritti in precedenza per cui il risultato finale non è dissimile da quanto sarebbe stato prodotto manualmente da uno sviluppatore. Alcuni dei limiti del processo automatico ad ogni modo emergono. Ad esempio tutti i campi delle classi di deserializzazione sono definiti di tipo stringa e questo dipende direttamente dallo schema che, essendo stato generato a sua volta in modo automatico dal file XML, non è stato in grado di determinare con precisione il tipo di ciascun nodo, limitandosi ad optare per un tipo stringa generico. È evidente che se nello schema fossero definiti con più precisione altri tipi per i nodi, il generatore automatico di sorgenti ne terrebbe conto. Pertanto si può concludere che il processo, sebbene automatico, può ricevere correzioni manuali che ne migliorini l'accuretezza e che questi interventi possono avvenire anche solo in alcuni punti e non dappertutto, lasciando automatiche le parti restanti del processo di generazione. Dunque, una volta ottenute queste classi, potrete persisterne delle istanze su un XML identico a quello di origine grazie al metodo *Serialize* dell'oggetto *XmlSerializer* e viceversa con il metodo *Deserialize*, esattamente come si è visto nella prima parte di questo articolo. Nei file allegati al presente articolo è presente anche il sorgente Visual Basic .NET generato automaticamente, nonché tutto il set di file per replicare quanto descritto finora. In realtà al parametro */classes* può essere sostituito il parametro */dataset* che impone la generazione non più di un sorgente *flat*, ma di un ben più complesso *typed dataset* e cioè di un dataset tipizzato secondo la struttura prescritta dallo schema. In realtà è proprio questo il processo che avviene *behind the scene* in Visual Studio .NET quando si genera un dataset tipizzato: l'IDE genera automaticamente uno schema XML a partire dalla query che popolerà il dataset tipizzato.

A questo punto *xsd.exe* genera la classe del *typed dataset* che diventa pronta all'uso. Tornando al nostro XML di esempio, un *typed dataset* basato su di esso ci consentirebbe di trattare le informazioni contenute nel XML come se fossero provenienti da un database e questo senza aver scritto nemmeno una riga di codice che ne consentisse quest'altra modalità d'uso.

***Fig. 2:** Lo schema xml appena generato visto da Visual Studio .NET*

CONCLUSIONI

Il sistema di serializzazione/deserializzazione nativo di .NET relativo al formato XML è davvero potente e di immediato utilizzo. Già le classi in grado di persistere una nostra classe su XML o di ricrearne le istanze a partire da questo ci consente di ottenere risultati evidenti a costo zero moltiplicando di fatto le possibilità di utilizzo di questa tecnica in diversi contesti applicativi. Ma l'utility *xsd.exe* ne arricchisce e completa le funzionalità portando l'uso di questa tecnologia ad un livello di semplicità e di potenza così sconcertante da fare eleggere, di fatto, questo tipo di tecnica come la via preferenziale in tantissimi ambiti.

Vito Vessia

Vito Vessia è cofondatore della codeBehind S.r.l. www.codeBehind.it, una software factory di applicazioni enterprise, web e mobile, dove progetta e sviluppa applicazioni e framework in .NET, COM(+) e Delphi occupandosi degli aspetti architetturali. È autore del libro "Programmare il cellulare", Hoepli, 2002, sulla programmazione dei telefoni cellulari connessi al PC con protocollo standard AT+. Può essere contattato tramite e-mail all'indirizzo vvessia@katamail.com

Utilizzare i template in Java con Velocity

Generazione di codice basata su template

Dopo una breve pausa torniamo ad affrontare la generazione automatica di codice. Mostreremo come progettare e sviluppare un generatore di codice in Java utilizzando il template engine Apache Velocity

Nei precedenti articoli (ioProgrammo n.79 e n.80) abbiamo visto come implementare un generatore di codice robusto e flessibile in Java. Come si ricorderà, il codice veniva generato dal modulo exporter in base alle informazioni presenti nell'*Internal Object Model*. La sintassi Java era memorizzata *(hard-coded)* direttamente nell'exporter. In quest'ultimo articolo dedicato alla generazione di codice vedremo come utilizzare i template nel processo di generazione, evidenziando i benefici derivanti da questa tecnica.

TEMPLATE E TRASFORMAZIONI

Il processo di trasformazione basato su template è molto diffuso nell'ambito dello sviluppo software. Generalmente, i componenti in gioco sono quattro:

- **Modello dati** - contiene le informazioni che devono essere trasformate;
- **Template** - rappresenta la struttura che i dati assumeranno alla fine della trasformazione; all'interno del template ci sono dei riferimenti ad informazioni presenti nel modello dati.
- **Template Engine** - è l'applicazione che, avendo in input il modello dati ed il template, effettua la trasformazione. Essa in pratica sostituisce i riferimenti interni al template con i dati veri e propri presenti nel modello.
- **Target** - è il risultato della trasformazione.

In Fig. 1 è schematizzata la relazione tra i quattro componenti.

Cerchiamo di chiarire il concetto con qualche esempio. Supponiamo di avere come modello dati il seguente file di testo:

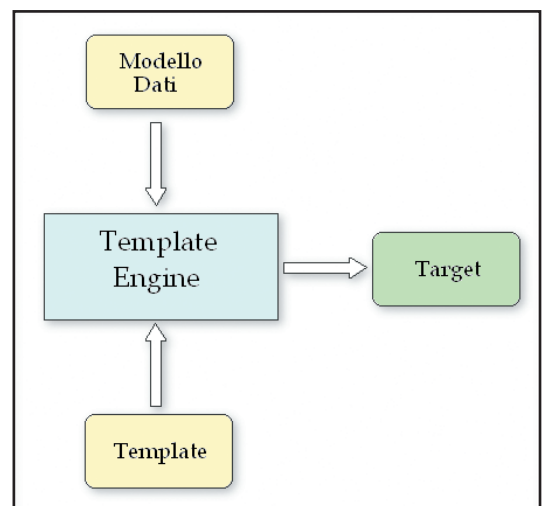



Fig. 1: *Relazioni tra i componenti di una trasformazione*

```
#persona.txt
$nome=Mario
$cognome=Rossi
```

Il template invece sarà il seguente file:

```
#persona.tmp
Ciao il mio nome è $nome e il mio cognome è $cognome
```

Come si può notare, nel template sono presenti le etichette *$nome* e *$cognome*. Esse rappresentano dei riferimenti alle medesime etichette del modello dati.

Supponiamo che l'applicazione trasforma sia il template engine che sostituisce i riferimenti nel template con i dati provenienti dal modello. Lanciamo adesso tale ipotetica applicazione passando in input il modello ed il template visti in precedenza:

```
> trasforma persona.txt persona.tmp
```

Il risultato della trasformazione (*target*) sarà:

Ciao il mio nome è Mario e il mio cognome è Rossi

Come si può notare, il template engine ha sostituito le etichette *$nome* e *$cognome* con i dati effettivi provenienti dal modello, ovvero Mario Rossi.
Se adesso utilizziamo come modello il seguente file:

```
#persona2.txt
$nome=Antonio
$cognome=Bianchi
```

Otterremo come target:

Ciao il mio nome è Antonio e il mio cognome è Bianchi

La situazione diventa ancora più interessante quando si cambia il template. Usiamo per esempio il seguente:

```
#persona2.tmp
**********************
Nome = $nome
Cognome = $cognome
**********************
```

Il risultato della trasformazione in questo caso sarà quindi:

```
**********************
Nome = Antonio
Cognome = Bianchi
**********************
```

Da questo si evince una condizione di notevole importanza: dati e template (ovvero la forma in cui vengono arrangiati) sono indipendenti.

GENERAZIONE DI CODICE BASATA SU TEMPLATE

È del tutto naturale pensare al processo di generazione di codice come ad un processo di trasformazione. Siamo in pratica in presenza di un modello dati, che conterrà le informazioni sulle entità che devono essere generate e di un template, che rappresenta la sintassi del linguaggio di programmazione target. Utilizzando il template engine descritto nel paragrafo precedente, possiamo costruire un banalissimo esempio che mostrerà i benefici nell'utilizzo dei template nel processo di generazione di codice.
Supponiamo che il nostro modello sia:

```
#studente.txt
$name=Studente
$base=Persona
```

e che invece il template sia:

```
#javaclass.tmp
public class $name extends $base
{
}
```

Il processo di trasformazione porterà quindi al seguente risultato:

```
public class Studente extends Persona
{
}
```

che è la definizione della classe *Studente* in Java.
Abbiamo quindi generato codice a partire dal modello dati e dal template. Per capire bene quali benefici si nascondono dietro i template, consideriamo il seguente:

```
#javainterface.tmp
public interface $name implements $base
{
}
```

ed utilizziamolo nel processo di trasformazione.
Otterremo:

```
public interface Studente implements Persona
{
}
```

In pratica abbiamo ottenuto un differente tipo di output cambiando solo il template e lasciando inalterati sia i dati sia, cosa più importante, il template engine, ovvero l'applicazione che effettua la trasformazione.
In questo caso, è stata generata un'interfaccia Java a partire dal modello dati che definisce *Studente* e *Persona*. Ma si può fare di più.
Consideriamo il seguente template:

```
#cpp.tmp
class $name : public $base
{
}
```

Effettuando la trasformazione otterremo:

```
class Studente : public Persona
{
}
```

Ovvero la definizione della classe *Studente* in C++!
Sempre senza toccare modello dati e template engi-

APACHE VELOCITY

La versione 1.4 del template engine Apache Velocity può essere scaricata all'indirizzo: http://jakarta.apache.org/velocity
Nel sito sono presenti il codice binario, i sorgenti ed una preziosa e dettagliata documentazione.
Per utilizzare Apache Velocity 1.4 all'interno delle vostre applicazioni Java, è necessario includere le librerie *velocity-1.4-rc1.jar* e *velocity-dep-1.4-rc1.jar* nella variabile d'ambiente *CLASSPATH*, sia in fase di compilazione, sia in esecuzione.

ne siamo addirittura riusciti a generare codice per un altro linguaggio!

APACHE VELOCITY

Generare codice utilizzando i template presuppone l'utilizzo di un template engine. Tale applicazione non è molto sofisticata, potrebbe esser un buon esercizio realizzarne uno fatto in casa. Attingere però a qualche prodotto open source già bello e fatto ci farà risparmiare sicuramente del tempo. Se programmate in Java, il template engine Apache Velocity risulterà un'ottima scelta.
Velocity, (disponibile per il download al link inserito a pag. 115), utilizza un semplice linguaggio proprietario per definire i template. Nel processo di trasformazione, il modello dati sarà composto da normali classi Java che verranno assegnate ad un determinato contesto. La trasformazione ha in input template e contesto e restituisce, in uno stream di output, la medesima struttura del template dove le etichette sono state sostituite con dati provenienti dal contesto. Gli esempi che seguiranno sono stati realizzati e testati utilizzando JDK 1.3.1, Apache Velocity 1.4 e Xerces-J 2.5.

NOTA

INVOCAZIONE DEI METODI GET.

All'interno di un template *Velocity* esistono due modi per invocare un metodo. Il primo è quello di specificare il nome del metodo così come è definito nella classe Java, per esempio:

```
$class.getName()
```

Il secondo metodo consiste nello specificare il nome del metodo senza get e senza parentesi:

```
$class.Name
```

Come si può notare, questa seconda forma è più compatta.

VELOCITY ALL'AZIONE

Supponiamo di voler generare del codice Java a partire da un modello dati in formato XML. L'esempio seguente mostra la struttura del modello:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- order.xml --!>
<Content>

  <Class name="Customer">
            <Attribute name="code" type="int"/>
    <Attribute name="description" type="String"/>
  </Class>

  <Class name="Order">
            <Attribute name="number" type="int"/>
    <Attribute name="date" type="Date"/>
    <Attribute name="customer" type="Customer"/>
  </Class>

</Content>
```

Come si può notare, l'elemento <*Class*> contiene l'attributo *name*, che rappresenta il nome della classe che deve essere generata, e un insieme di elementi <*Attribute*>, che sono i dati membro per quella classe. Quest'ultimo elemento contiene gli attributi *name* e *type*, rispettivamente il nome e il tipo del dato membro. Nell'esempio sono state definite due classi: *Order* e *Customer*. In base a tali informazioni vorremmo che il nostro generatore generi le classi Java *Order* e *Customer* che hanno i dati membro descritti all'interno del documento XML e i metodi *getXxx* e *setXxx* per accedervi.
Il primo passo è creare una struttura interna in grado di ospitare i dati. In pratica, vista la loro semplicità, ciò consiste nel definire due classi, una per l'elemento <*Class*> e l'altra per l'elemento <*Attribute*>, che chiameremo descrittori. Essi conterranno i dati provenienti dal file XML.
Ecco l'implementazione dei due descrittori:

```
// ClassDescriptor.java
package com.codegenerator.example1;
import java.util.*;
public class ClassDescriptor
{
  private String name;
  private ArrayList attributes = new ArrayList();
  public void setName(String name)
  {
     this.name = name;
  }
  public String getName()
  {
     return this.name;
  }
  public void addAttribute(AttributeDescriptor attribute)
  {
     attributes.add(attribute);
  }
  public ArrayList getAttributes()
  {
     return attributes;
  }
}

// AttributeDescriptor.java
package com.codegenerator.example1;
import java.util.*;
public class AttributeDescriptor
{
   private String name;
   private String type;
   public void setName(String name)
   {
       this.name = name;
   }
   public String getName()
   {
      return this.name;
   }
   public void setType(String type)
   {
      this.type = type;
   }
```

```
  public String getType()
  {
      return this.typ e;
  }
}
```

Per importare i dati dal documento XML ai descrittori useremo un parser SAX, come mostrato dal codice seguente:

```
package com.codegenerator.example1;
import java.util.*;
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
public class ClassDescriptorImporter extends
                                                    DefaultHandler
{
  private ArrayList classes = new ArrayList();
  public ArrayList getClasses()
  {
     return classes;
  }
  public void startElement(String uri, String name,
    String qName, Attributes attr) throws SAXException
  {
    if (name.equals("Class"))
    {
      ClassDescriptor cl = new ClassDescriptor();
      cl.setName(attr.getValue("name"));
      classes.add(cl);
    }
    else if (name.equals("Attribute"))
    {
      AttributeDescriptor at = new AttributeDescriptor();
      at.setName(attr.getValue("name"));
      at.setType(attr.getValue("type"));
      ClassDescriptor parent =
          (ClassDescriptor) classes.get(classes.size()-1);
      parent.addAttribute(at);
    }
    else if (name.equals("Content"))
    {
    }
    else throw new SAXException("Element " + name +
                                              " not valid");
  }
  public void endElement(String uri,
    String name, String qName) throws SAXException
  {
  }
}
```

La classe *ClassDescriptorImporter* ha il compito di trasferire i dati provenienti da un documento XML nei descrittori *ClassDescriptor* e *AttributeDescriptor*. Come si può notare, ogni volta che s'incontra un elemento *<Class>*, allora un nuovo *ClassDescriptor* viene creato ad aggiunto all'*ArryList classes*. Quando è invece un elemento *<Attribute>* ad essere processato, allora un nuovo *AttributeDescriptor* viene creato ad aggiunto all'oggetto *ClassDescriptor* parent. Alla fine, nell'*ArrayList classes*, avremo i descrittori relativi a tutti gli elementi *<Class>* presenti nel documento XML in input. Il metodo *getClasses* restituirà appunto tale ArrayList. Adesso entra in gioco *Velocity*. Il template per generare una classe Java con attributi e relativi metodi *getXxx* e *setXxx* è il seguente:

```
## class.vm
import java.util.*;
public class $class.Name
{
  #foreach( $att in $class.Attributes)
      // $att.Name
      private $att.Type $att.Name;
      public $att.Type
              get$utility.firstToUpperCase($att.Name)()
      {
          return this.$att.Name;
      }
      public void set$utility.firstToUpperCase(
                      $att.Name)($att.Type $att.Name)
      {
          this.$att.Name = $att.Name;
      }
#end
}
```

L'etichetta *$class* non è altro che un riferimento al nostro *ClassDescriptor*. Quindi *$class.Name* è il risultato dell'invocazione del metodo *ClassDescriptor.getName()*. L'istruzione *#foreach* itera il blocco di codice successivo per tutti gli elementi restituiti da *$class.Attributes*, che è il risultato dell'invocazione del metodo *ClassDescriptor.getAttributes()*. In questo blocco vengono definiti gli attributi ed i metodi *getXxx* e *setXxx* in base al valore presente in *$att.Name* e *$att.type*, che sono rispettivamente il risultato dell'invocazione di *AttributeDescriptor .getName()* e *AttributeDescriptor.getType()*. L'etichetta *$utility.firstToUpperCase* richiama un metodo definito dal programmatore (lo vedremo in seguito) che trasforma il primo carattere di una stringa in maiuscolo. Tale metodo è utile per ottenere, per esempio, *getNumber* a partire dall'attributo number. Quello che rimane da implementare è l'applicazione principale, cioè il generatore, che legge il modello dati dal documento XML (utilizzando la *ClassDescriptorImporter*), lo associa al template appena visto e da inizio alla trasformazione utilizzando *Velocity*. Ecco il codice relativo al metodo *start* del nostro generatore (il listato completo può essere reperito nel CD allegato alla rivista):

```
public static void start(String modelFile, String
```

• SCRIVERE UN GENERATORE DI CODICE IN JAVA
G. Naccarato
ioProgrammo
2004

• IMPLEMENT AN IOM-BASED CODE GENERATOR
G. Naccarato
JavaPro Online
Marzo 2004
www.ftponline.com/javapro/2004_03/online/j2ee_gnaccarato_03_10_04/

• CODE GENERATION IN ACTION
Jack Herrington
Manning
2003
www.manning.com/herrington/

• CODE-GENERATION TECHNIQUES FOR JAVA
Jack Herrington
OnJava
Settembre 2003
www.onjava.com/pub/a/onjava/2003/09/03/generation.html?page=1

```
                                        templateFile)
 throws Exception
 {
    // Importa il modello dati XML
    FileInputStream input = new
                              FileInputStream(modelFile);
    xmlReader.parse(new InputSource(input));
    input.close();
    classes = cdImporter.getClasses();
                                  // ClassDescriptor Array
    //Genera le classi usando Velocity
    GeneratorUtility utility = new GeneratorUtility();
    for (int i = 0; i < classes.size(); i++)
   {
     VelocityContext context = new VelocityContext();
     ClassDescriptor cl = (ClassDescriptor) classes.get(i);
     context.put("class", cl);
     context.put("utility", utility);
     Template template = Velocity.getTemplate(
                                          templateFile);
     BufferedWriter writer = new BufferedWriter(
                   new FileWriter(cl.getName()+".java"));
     template.merge(context, writer);
     writer.flush();
     writer.close();
     System.out.println("Class " + cl.getName() +
                                        " generated!");
 }
}
```

SUL WEB

Apache Velocity
http://jakarta.apache.org/velocity

Code Generation Network
www.codegeneration.net

Giuseppe Naccarato
www.giuseppe-naccarato.com

Il metodo ha in input il nome del file XML e quello del template. Esso importa i dati presenti nel file XML utilizzando l'oggetto *xmlReader* precedentemente associato all'oggetto cdImporter, che è un istanza di *ClassDescriptorImporter.* Da tale oggetto si ottiene la lista dei descrittori delle classi da generare *(ArrayList classes)*. A questo punto, all'interno del ciclo *for,* viene creato un contesto per Velocity *(context)*. Esso è importantissimo, poiché di fatto lega i descrittori al template. Il metodo *put* associa una classe Java ad un'etichetta di Velocity. Infatti, scrivere:

```
context.put("class", cl);
context.put("utility", utility);
```

significa che l'oggetto *cl* (descrittore della classe attualmente processata) sarà riferito all'interno del template dall'etichetta *$class* e l'oggetto utility dall'etichetta *$utility.* Quest'ultimo oggetto è un'istanza della classe *GeneratorUtility* che contiene il metodo *firstInUpperCase.* A questo punto, viene creato un oggetto *Template* a partire dal file template in input. Il metodo *merge* è l'autore materiale della trasformazione. Esso prende i dati dal contesto (context) e crea l'output *(writer)* in base alle direttive presente nel template.
Lanciando il generatore con in input *order.xml* e *class.vm* otterremo i file *Order.java* e *Customer.java.* Ecco il codice generato relativo al primo:

```
import java.util.*;
public class Order
{
  // number
  private int number;
  public int getNumber()
  {
    return this.number;
  }
  public void setNumber(int number)
  {
    this.number = number;
  }
  // date
  private Date date;
  public Date getDate()
  {
    return this.date;
  }
  public void setDate(Date date)
  {
    this.date = date;
  }
  // customer
  private Customer customer;
  public Customer getCustomer()
  {
    return this.customer;
  }
  public void setCustomer(Customer customer)
  {
    this.customer = customer;
  }
}
```

Come si può facilmente intuire, senza toccare i dati e senza metter mano al generatore, potremmo generare un input differente semplicemente modificando il template. Questa caratteristica rende il generatore di codice notevolmente flessibile.

CONCLUSIONI

I concetti esposti in quest'articolo possono essere usati per implementare un generatore basato su un Internal Object Model, come visto il mese scorso, che usa template. Quello che si deve fare è creare un exporter che presenti la sintassi del linguaggio target all'interno del codice (hard-coded), bensì in uno o più template. Dall'IOM sarà possibile estrarre le informazioni per creare un contesto su cui Velocity potrà operare. Purtroppo quest'aspetto sarà lasciato per esercizio. Buon divertimento.

Giuseppe Naccarato

Applicazioni ludiche e trattamento di griglie esagonali

Nuove simulazioni per griglie esagonali

Sistemi di numerazione basati su proiezioni isometriche cubiche, in spazi tridimensionali, rendono più facile ed efficiente il trattamento delle informazioni nelle griglie esagonali.

Circa le numerose e utili applicazioni delle griglie esagonali, sia in campo strategico-militare che in ambienti ludico-scientifici, abbiamo già discusso, scoprendo che rivestono un ruolo considerevole. Una domanda però non ha ancora trovato risposta ad un mese dalla scrittura del primo articolo sull'argomento. Mi chiedo come mai le api, gli insetti per intenderci, abbiano scelto come elemento base per il loro alveare proprio l'esagono. Non nego che, a tal proposito, ho effettuato delle ricerche, i risultati non sono però soddisfacenti. Penso che la domanda sia lecita anche per un programmatore, non per l'aspetto etologico, che è sicuramente interessante ma esula dagli scopi della nostra rivista, bensì per ciò che riguarda un argomento a noi molto caro, la logistica. Una risposta plausibile sulla scelta dell'esagono riguarda la naturale propensione che ha questa figura per la costruzione di griglie. L'esagono, il quadrato (anche il rettangolo) e il triangolo, sono le sole figure che possano dare origine a griglie regolari. In strutture esagonali però, ogni elemento "comunica" con ben sei celle. Inoltre, il contatto con gli elementi adiacenti avviene solo mediante il lato e non attraverso gli spigoli. Nella scacchiera, invece, una casa comunica con i quattro lati del quadrato, anche se vi sono spigoli in comune con ulteriori quattro elementi (proprietà non molto gradita in alcuni modelli). In definitiva, possiamo dire che l'esagono ben si presta alla mappatura di strutture regolari. Nel presente articolo, come la scorsa volta, seguiremo una logica del doppio binario. Tratteremo da un lato nuovi metodi per la numerazione delle celle all'interno di griglie, e vedremo quali sono i possibili sviluppi; mentre sull'altro binario verrà approfondito il gioco *Hex*, così, oltre a divertirci, esploreremo utili percorsi per la manipolazione degli elementi oggetto dei nostri studi. Cominciamo dando sfogo al divertimento, trattiamo Hex. Del resto il proverbio dice: prima il piacere e dopo il dovere… ah! non è così? Fa lo stesso. Cominciamo.

I FIGLI DI HEX

Come introduzione, ripeto in estrema sintesi le regole del gioco: sono coinvolti due giocatori associati a due colori, solitamente si usano rosso e blu. Il campo di gioco, manco a dirsi, è una griglia esagonale a forma di rombo, in cui ogni lato conta undici esagoni. Lati opposti hanno colori uguali. In *Fig. 1a* è mostrata la scacchiera. A questo proposito, consentitemi la leggera imprecisione: a rigore, non si potrebbe parlare di scacchiera, ma risulterebbe noioso e stucchevole il continuo riferimento alla *"griglia esagonale"* o al *"campo di gioco"*.

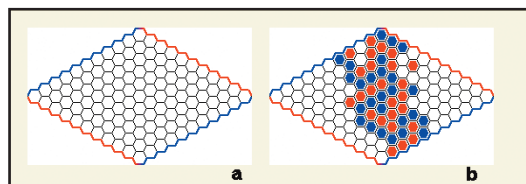



Fig. 1: **a) Campo di gioco; b) Esempio di partita, qui vince il blu**

I contendenti a turno pongono una tessera del proprio colore sulle celle vuote o se preferite colorano un esagono vuoto. Scopo del gioco è formare una catena del proprio colore che tocchi i due lati. L'obiettivo secondario è evitare che l'avversario costruisca la propria catena e vinca. Questo secondo obiettivo risulta maggiormente importante se si considera che in Hex una partita non può terminare in parità. L'analisi, che proporrò in questo numero riguardo al gioco, è la dimostrazione della precedente affermazione. L'enunciazione è: al termine della partita, uno fra il giocatore blu e il giocatore rosso risulterà vincitore. In *Fig. 2b* è proposta una partita nello stadio finale. Per effettuare la dimostrazione è necessario conoscere un altro gioco. A tale scopo,

ma anche per piacere, ci concederemo una parentesi sui giochi che derivano da Hex: quelli che ho battezzato *i figli di Hex*. Onore inizialmente ad un illustre personaggio, uno studioso a cui l'informatica deve molto, Claude Shannon; egli formulò le regole del gioco omonimo. Nel *gioco di Shannon* si ha come campo di gioco un grafo, così come mostrato in *Fig. 2a*, costituito da due vertici chiamati *s* e *t*.
I due giocatori, generalmente chiamati *short* e *cut* (gli "italianofili" possono tradurre a piacimento) a turno colorano uno dei vertici del grafo. L'obiettivo di short, al fine di vincere la partita, è quello di creare una catena che colleghi i due vertici *s* e *t*. Dal canto suo cut per vincere deve impedire a short di raggiungere il suo obiettivo. Hex risulta uno speciale caso del gioco di Shannon.

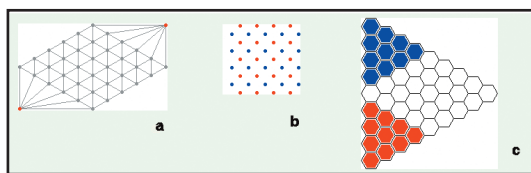



Fig. 2: ***a) Grafo di Shannon di dimensione 6; b) Campo di gioco per bridg-it; c) gioco Y***

Un altro gioco introdotto in seguito all'uscita di Hex è *bridg-it*. Questa volta l'arena di gioco è una serie di punti sfalsati e colorati (manteniamo la convenzione con il rosso e il blu) in modo alternato, come si può vedere in *Fig. 2b*. I due giocatori devono tracciare dei segmenti (piccoli ponticelli, da cui *bridge*) che uniscano due propri punti adiacenti, allo scopo di unire con una catena di segmenti le due facce esterne del proprio colore. Ovviamente, un segmento non si può intersecare con niente. Ricompaiono le griglie esagonali per il *gioco Y*. Qui siamo di nuovo di fronte a una generalizzazione di Hex. Vi consiglio di porre particolare attenzione a tale gioco, poiché ce ne serviremo nel prossimo paragrafo per la dimostrazione che vi ho accennato. Il campo di gioco è una griglia triangolare sulla quale a turno i due giocatori hanno il solito compito di collocare le proprie tessere. Lo scopo del gioco, identico per i due partecipanti, è quello di formare una catena che tocchi tutti e tre i lati della scacchiera. Se i due giocatori nella prima fase della partita colorano due triangoli in modo da lasciare una griglia a forma di rombo, allora gioco *Y* degenera in Hex; da cui si deduce che Hex è appunto uno speciale caso di *gioco Y*. In Fig. 2 c è mostrata una fase della partita a *gioco Y* nella particolare situazione appena descritta. Per terminare la rassegna vi presento *twixt*. Si tratta di una variante di bridg-it, infatti, usa lo stesso campo di gioco. I giocatori hanno a disposizione una nuova mossa, possono tracciare segmenti tipici del cavallo negli scacchi, con la possibilità di scavalcare i tracciati avversari. Al contrario di tutti i giochi finora esaminati in twixt si può avere la patta; non so se sia una cosa positiva o negativa, giudicatelo voi; per me questa possibilità rende più ricco un gioco. Ad ogni modo dopo un'analisi complessiva e avendo provato alcuni dei giochi descritti, ritengo il più giocabile proprio Hex. Questione di gusti.

O SI VINCE O SI PERDE

Giocando a Hex la sensazione che o si vinca o si perda, si avverte sin dal principio. Ma che tale affermazione sia dimostrabile è tutt'altra cosa. Niente paura, anche essendo un teorema, con tanto di ipotesi, tesi e dimostrazione, è proprio divertente, e ci aiuterà a conoscere meglio le griglie esagonali. La dimostrazione si basa sul gioco *Y*, ma poiché esso è una generalizzazione di Hex, allora, vale anche per questo ultimo. Il teorema afferma: una partita a gioco *Y* non può terminare mai in pareggio. Sarà sufficiente dimostrare quanto segue: se tutte le celle sono colorate blu o rosse allora il vincitore sarà il blu o il rosso. Questo asserto si comprenderà meglio alla fine della disquisizione. La dimostrazione si basa sul concetto di riduzione. Una configurazione può essere, infatti, ridotta di una unità. Cerchiamo di capire il procedimento mediante un esempio. Consideriamo una configurazione di gioco *Y* di dimensione 6, come riportato in Fig. *3a*. Il procedimento di riduzione valuta piccoli gruppi di tre celle adiacenti che formino un triangolo e che abbiano lo stesso orientamento dell'intero triangolo campo di gioco. Nel processo di riduzione ogni triangolo viene sostituito da una cella. Il colore della nuova cella è quello che predomina nel triangolino (che si presenta in maggioranza). Prima considerazione: la configurazione ridotta non diventa di dimensione pari a un terzo della iniziale poiché uguali celle faranno parte di differenti triangolini. Secondariamente, non si presentano mai incongruenze, ossia, i colori delle celle nella configurazione ridotta sono sempre univocamente definiti. In *Fig. 3*, da *b* a *f*, si possono osservare le successive riduzioni che hanno portato una configurazione di gioco *Y* da dimensione 6 a 1. La notizia sorprendente è il risultato del seguente lemma: il colore della configurazione ridotta di dimensione 1 indica il vincitore della partita a gioco *Y*. La dimostrazione è semplice. Formalizzata una catena come una sequenza o stringa di esagoni dello stesso colore, la dimostrazione si basa su osservazione cruciale: dopo la riduzione una catena viene preservata. Lo si può notare dall'esempio; ma si ha una spiegazione rigorosa notando che ogni collegamento della catena corrisponde a un piccolo triangolo in cui almeno due celle sono del colore della catena. Una catena vincente genera sempre una catena vincente dopo la riduzione, includendo anche l'ultima che a grandezza degenere pari a 1. Se un giocatore deve

JOHN NASH

Il premio Nobel, nonché soggetto del film "A beautiful mind" ha introdotto il gioco Hex. Pare che sia stato ispirato dalle mattonelle esagonali presenti in un bagno della sua università. Egli battezzò il gioco Nash. Contemporaneamente, anche in Danimarca Piet Hein scopriva lo stesso gioco a cui dava il nome Polygon.

necessariamente vincere è provato che non vi siano possibilità di patta. Notevole vero? Adesso, si comprende anche la formulazione dell'ipotesi e della tesi: se tutte le celle sono colorate blu o rosse allora il vincitore sarà il blu o il rosso.
Esiste una bella dimostrazione, anche se più articolata, di David Gale elaborata direttamente sulla scacchiera di Hex.

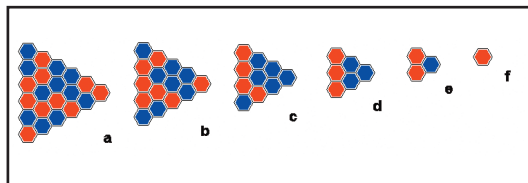



Fig. 3: **a) Configurazione iniziale –dimensione 6; b) riduzione a 5; c) riduzione a 4; d) riduzione a 3; e) riduzione a 2; f) riduzione a 1**

CIRCA LA STRATEGIA

Il primo libro apparso sul mercato che affronta il problema della strategia in Hex è Making the right connection scritto da Cameron Browne.

PROIEZIONI ISOMETRICHE

Passiamo ad un trattamento più tecnico delle griglie. Nello scorso appuntamento abbiamo proposto una possibile numerazione. Ad ogni cella, in quella occasione, abbiamo associato una coppia di numeri secondo una logica bidimensionale. Il primo numero indicava la sua posizione lungo l'asse delle ascisse, mentre, il secondo la posizione lungo quello delle ordinate. Una tale numerazione è sicuramente semplice ed intuitiva, pone però dei limiti qualora si vogliano sviluppare particolari funzioni di percorrenza e manipolazione delle griglie stesse. A tale proposito è stato rilevato come numerazioni dei singoli esagoni con tre cifre, anche se generano una ridondanza, peraltro non fastidiosa poiché si tratta di un semplice numero, consentano una maggiore flessibilità nel trattamento delle informazioni. In Fig. 4 è presentato tale numerazione.

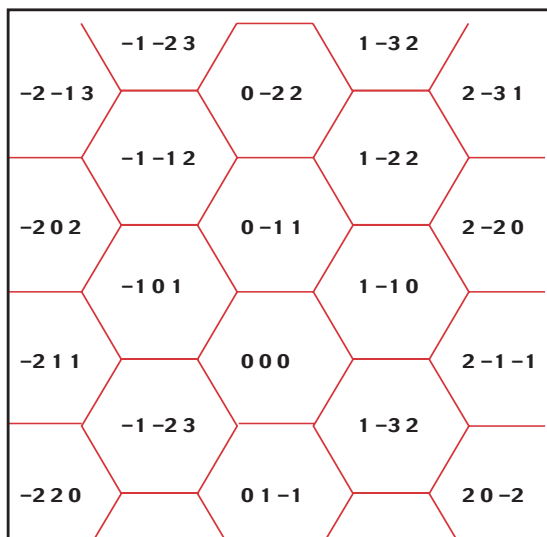



Fig. 4: **a) Griglia esagonale numerata mediante terne**

Prima di entrare nei particolari e capire con quale procedimento sia stata ottenuta, diamo una occhiata generale per notare le caratteristiche macroscopiche. Il primo numero della terna ha un significato immediato, rimane uguale per bande verticali, e cresce di uno verso destra, mentre decresce verso sinistra. Come vedremo è associato all'asse *x*. Il secondo numero rimane, invece, constante lungo catene di esagoni obliqui, in particolare verso la direzione nord-est o se preferite sud-ovest. Analogo discorso vale per il terzo parametro nella direzione nord-ovest, sud-est. Un ulteriore sguardo alla griglia e si nota che per ogni cella la somma dei tre numeri da sempre zero. Tutte caratteristiche che lasciano presagire una agevole manipolazione. Ma facciamo un passo indietro, affrontiamo il procedimento per la costruzione di tali griglie con riferimento a un sistema di tre assi cartesiano simmetrico. La simmetria scaturisce dalla scelta di un angolo uguale tra i vari assi, che non può essere che 120°.

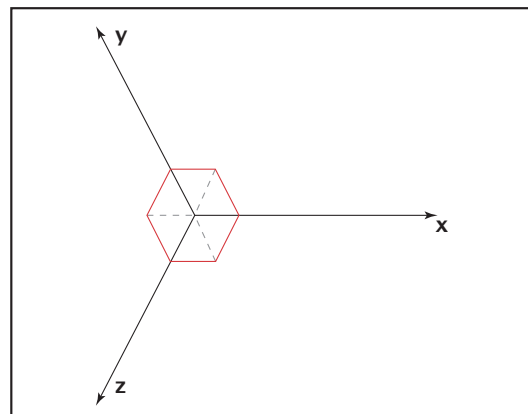



Fig. 5: **Sistema tridimensionale simmetrico. Proiezione cubica isometrica**

Come mostrato in Fig. 5, *x* cresce verso destra, *y* aumenta verso sopra a sinistra e *z* in basso a destra. Se su questo sistema di assi produciamo una proiezione cubica isometrica otterremo le griglie tanto agognate. Una proiezione isometrica è una proiezione ortografica nel piano $x+y+z=0$. È questo lo standard dei disegnatori. Se, come fatto in *Fig. 5,* su un sistema di assi così costruito disegniamo un cubo, le proiezioni sui tre piani del vertice opposto all'origine descriveranno proprio un esagono. Mi rendo conto che il cubo costruito con questa prospettiva, sia inizialmente difficile da vedere, consiglio di immaginare un cambio di prospettiva (immaginate una piccola rotazione degli assi). Affiancando più cubi, le relative proiezioni dei vertici produrranno la griglia. Adesso, si presenta il problema di trasformare una terna di coordinate presenti nello spazio tridimensionale descritto in un esagono secondo la griglia costruita (*Fig. 4*). Bisogna discretizzare i punti. Eventuali terne reali devono diventare terne di numeri interi. La formula di riferimento per la trasformazione è riportata di seguito. Nel prossimo paragrafo svilupperemo l'implementazione.

$$\overline{rin}(\overline{rin}(\overline{r}) - (0.5 - max_i \mid rin(r_i) - r_i \mid) \textstyle\sum_i rin(r))$$

Il vettore *r* (con il simbolo di overscore) indica la posizione iniziale che non è stata discretizzata, e quin-

di non associata ad un esagono della griglia; si tratta quindi dell'input. L'output, ovvero il risultato della formula è l'esagono contenente il punto secondo la numerazione convenuta. La funzione *rin* restituisce semplicemente la parte intera di un numero reale, mediante arrotondamento; se applicata a un vettore restituisce il vettore delle parti intere delle singole componenti. *max* calcola il massimo degli scarti in valore assoluto, tra valore reale e valore arrotondato di ogni componente del vettore. Applicare tale formula equivale a verificare che il punto sia interno al cubo.

IL CODICE

Un'efficiente implementazione della formula è approntata con il seguente codice C++.

```
..
/* x,y e z sono le componenti iniziali */
double rx, ry, rz // variabili di lavoro
int ix, iy, iz, s;
ix = rx = rin(x);
iy = ry = rin(y);
iz = rz = rin(z);
s= ix+iy+iz;
if(s) { double abs_dx=rabs(rx-x), double
        abs_dy=rabs(ry-y), double abs_dz=rabs(rz-z);
  if (abs_dx>=abs_dy && abs_dx>= abs_dz) ix-=s;
                              // abs_dx è il maggiore
  else if(abs_dy>=abs_dx && abs_dy>= abs_dz)
                         iy-=s; // abs_dy è il maggiore
  else iz-=s;
}
/* ix, iy, iz sono le nuove coordinate della griglia
                                            esagonale
..
```

Nell'assegnazione delle componenti da aggiornare, in funzione dell'individuazione del massimo scarto, al vecchio valore viene sottratto *s* (somma delle componenti), in conformità con la formula; allo scopo viene usato l'operatore automatico C++ -=. Si può notare che l'algoritmo è simmetrico in *x, y* e *z,* giacché, gli assi sono tra loro simmetrici. Inoltre, la rilevante proprietà che vede la somma ix+iy+iz pari a zero rende il tutto più flessibile. Ad esempio, se si tiene (o se vi sono delle esigenze specifiche) al risparmio di memoria si può evitare di mantenere *z* in RAM calcolandolo di volta in volta. E così si possono fare tante utili modifiche in funzione delle esigenze del proprio programma. L'ottimizzazione del codice dipende da diversi fattori, ad esempio, da come vengono svolte le operazioni di arrotondamento e valore assoluto (rin e rabs). Per tale motivo si demanda l'implementazione allo sviluppatore, che sfruttando al meglio le caratteristiche del compilatore potrà codificare al meglio queste parti. Un consiglio è che *rin* e *rabs* siano funzioni in linea.

CONCLUSIONI

A partire da un sistema siffatto si possono introdurre nuove funzioni. Sicuramente, si renderà necessaria una trasformazione dalle coordinate del mouse, prodotte su un sistema di coordinate bidimensionali, al sistema proposto tridimensionale. Per farlo si possono adottare le più volte viste matrici di trasformazione lineare. Nel caso specifico è comunque possibile estrarre semplici formule, per ottenere le trasformazioni nei due sensi. Poiché il sistema 3D è simmetrico le espressioni per *y* e *z* sono speculari, inoltre, la *x* rimane sostanzialmente la stessa. Come dicevo, tali strutture sono la base per ulteriori studi. Ad esempio, è stata introdotta la distanza, e la distanza di percorrenza. La prima è la mera applicazione del teorema di Pitagora e rappresenta la distanza euclidea, ossia la lunghezza del segmento che unisce i due esagoni. Se i due esagoni sono *(x1, y1,z1)* e *(x2,y2,z2)* la formula è:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Mentre la distanza di percorrenza tra due esagoni *(x1,y1,z1)* e *(x2,y2,z2)*, indica il numero di celle che bisogna percorrere per passare dall'uno all'altro.
La formula è:

1/2[|x1-x2|+|y1-y2|+|z1-z2|]*

Un altro metodo per il calcolo della distanza di percorrenza prevede l'individuazione di un massimo tra i valori assoluti delle singole componenti, è un po' bizzarro ma funziona. Ecco come si può esprimere in forma sintetica.

max[|x1-x2|+|y1-y2|+|z1-z2|]

Insomma, abbiamo introdotto nuovi spunti di analisi e discussione circa le interessantissime griglie esagonali. Certo, ci sarebbe ancora molto da dire, e probabilmente in un futuro non molto lontano lo faremo. Merita sicuramente approfondimento l'aspetto strategico di Hex, così come risulterebbe utile sviluppare nuove funzioni per le griglie. Chi vuole può spaziare in molte direzioni. Infine, ricordo che sebbene gli ambiti di utilizzazione di tali strutture siano molti, non esiste una teoria organica sull'argomento. Può essere questo ultimo un invito a chiunque abbia voglia e tempo per approfondire seriamente la questione. Io vi saluto e vi aspetto per il prossimo appuntamento. A proposito chiunque mi possa dare delucidazioni sulle api mi faccia sapere.

Fabio Grimaldi

NOTA

David Gale, studente di John Nash, ha introdotto bridg-it. Gioco Y è opera di Claude Shannon e C. Schensted e C. Titus. Infine, A. Randolf ha ideato twixt.

Gioco basato sulle proprietà e le operazioni dei numeri interi

Il gioco del divisore

Il gioco del divisore mette alla prova le nostre abilità tattiche sulla base di una semplice operazione sui numeri interi Ancora una volta i numeri sono i protagonisti dei nostri enigmi

Conoscenze richieste
Algebra elementare, basi di C++, nozioni di programmazione strutturata

Software
Nessuno

Impegno

Tempo di realizzazione

Chi mi conosce sa bene che sono appassionato di ogni genere di rompicapo, che si svolga su una scacchiera o che impieghi dischi e paletti. Ma nell'ambito di questa passione, che da tempo coltivo e che cerco di condividere con voi, emerge una predilezione: i giochi e gli enigmi che fanno uso di semplici elementi, i più semplici possibile, ossia numeri e lettere. In questo contesto abbiamo già esaminato alcuni enigmi come quello storico di Euclide. È adesso la volta del gioco del divisore, ovvero, un modo per divertirsi con gli amici avendo a disposizione anche solo carta e penna.

IL GIOCO DEL DIVISORE

L'enigma, conosciuto anche come gioco delle aliquote, si basa, come si può intuire, sul concetto di divisore. Sperando di non ledere la sensibilità di nessuno, ricordo il concetto di divisore. Un numero *X* è divisore di un numero *Y* se *Y* è divisibile per *X*, o espresso diversamente se il resto della divisione intera tra *Y* e *X* è zero. Facciamo qualche esempio, i divisori del numero 12 sono: 1, 2, 3, 4, 6 e 12. Mentre, i divisori di 11 sono 1 e 11. Un numero che abbia come divisori solo 1 e se stesso è detto primo. Per i programmatori (Pascal in particolare) possiamo dire che *X* è un divisore di *Y*, se *Y mod X=0*. L'operatore *mod* restituisce il resto di una divisione intera. La funzione riportata di seguito calcola la somma dei divisori di un numero, eccetto se stesso.

```
(* Funzione somma dei divisori eccetto se stesso *)
function divisori(p1:longint):longint;
var i,somma:longint;
begin
  somma:=0;
  for i:=1 to p1 div 2 do
    if p1 mod i = 0 then somma:=somma + i;
  divisori:=somma;
end;
```

Passiamo al gioco. La competizione è tra due. Dopo aver scritto su un foglio un numero iniziale, che può essere generato casualmente (estratto con i classici bigliettini, o più semplicemente il numero di pagina di un libro aperto a caso), o deciso da uno dei due giocatori (in questo caso bisogna a ogni partita alternare il giocatore che lo propone), bisogna a turno produrre a partire dal numero uno nuovo. La prima mossa prevede di scrivere sul foglio un nuovo numero. Il nuovo numero si ottiene dalla differenza tra il vecchio e un suo divisore (elemento di scelta del giocatore), non sono considerati divisori validi, l'uno e il numero stesso. Così, si procede alternativamente nella produzione di nuovi numeri. Perde chi non riesce a sottrarre alcun numero, ovvero, chi si trova davanti un numero primo. L'altro giocatore sarà, ovviamente, il vincitore. In definitiva, l'obiettivo del giocatore è lasciare l'avversario senza mosse. Il gioco è estremamente semplice, è pero necessario individuare una corretta strategia per la vittoria, in considerazione del fatto che se il numero iniziale è grande, allora sarà necessario adottare una congrua scelta. La costruzione di una strategia vincente è la sfida che lancio questa settimana. Per cui buon lavoro.

NUMERI PERFETTI

Non è vero che tre sia un numero perfetto, o almeno non è vero se facciamo riferimento alla definizione matematica di numero perfetto. Così, scopriremo che 6 è perfetto e non 3; non solo, scopriremo interessanti curiosità storiche. Un numero è perfetto se la somma dei suoi divisori eccetto se stesso è pari proprio al numero stesso. Il più piccolo numero perfetto è il 6, infatti, sommando i suoi tre divisori, 1, 2 e 3 si

ottiene proprio 6. Questa come altre curiosità numeriche trova la sua origine negli studi dei pitagorici del VI secolo a.C., ma riferimenti ai numeri perfetti si trovano sparsi nel corso dei secoli, ed è questo un fattore di interesse e mistero. Nell'opera *"Creazione del mondo (III)"* il filosofo ebreo del primo secolo Philo Judaeus scrive che Dio, riconoscendo la perfezione intrinseca del 6, volle creare il mondo con questo numero di giorni. Il concetto fu ripreso e confermato da Sant'Agostino nella sua opera *"Città di Dio"*. Altri numeri perfetti, che si possono individuare eseguendo la funzione pascal che ho costruito sono: il 28, il 496 e l'8128 che erano già conosciuti dai Greci. Per individuare il quinto dei numeri perfetti bisogna contare parecchio, fare molte divisioni e somme, far girare molto il microprocessore, infatti, si tratta di una cifra di un certo "peso" 33550336, sicuramente non snella vero? La cosa sbalorditiva è che quest'ultimo numero era conosciuto già nel XV secolo, presso i tedeschi, è infatti reperibile in un codice del tempo; se non ricordo male all'epoca non si poteva fare affidamento su elaboratori, ne tanto meno su calcolatrici tascabili. A tutt'oggi, nell'era digitale, i numeri perfetti conosciuti sono all'incirca quaranta. La funzione che ne verifica la perfezione è riportata di seguito, si avvale della routine che calcola la somma dei divisori, e semplicemente impone la condizione di eguaglianza tra questa e il numero passato come parametro. L'eguaglianza, ovviamente, indica che il numero è perfetto.

```
function perfetto(p1:longint):boolean;
begin
   perfetto:= p1=divisori(p1);
end;
```

NUMERI AMICI

Una variante analoga al numero perfetto che però coinvolge due numeri interi è quella conosciuta come numeri amici. Contrariamente a quanti pensano che i numeri siano semplici sequenze di cifre senza sentimenti, si sappia che è possibile avere un legame di amicizia anche tra due numeri. Si ha quando la somma dei divisori dell'uno è pari all'altro e contemporaneamente la somma dei divisori dell'altro è pari al primo. Sono amici, ad esempio, i due numeri 220 e 284. Infatti, per 220 i divisori sono 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110 la cui somma è 284 e analogamente si prova l'altra eguaglianza. Anche per questi numeri è stato ritrovato un riscontro storico.
Nel libro di Martin Gardner "Show di magia matematica" si parla di talismani venduti nel Medioevo che riportavano incisi tali numeri, indossarli favoriva un vincolo sentimentale. Ma anche presso gli arabi si usava incidere su frutti le due cifre in questione e offrire all'amante il frutto così trattato come afrodisiaco. La funzione da me sviluppata per verificare questo legame tra due numeri fa uso della funzione divisori e attraverso un'espressione con due condizioni relazionate da and verifica appunto la caratteristica prima descritta.

```
(* Funzione per il calcolo di coppie di numeri amici *)
function amici (p1,p2:longint):boolean;
begin
   amici:= (divisori(p1)=p2) and (divisori(p2)=p1);
end;
```

NOTA

PRIMI DI MERSENNE

Dagli studi di Euclide ed Eulero svolti a secoli di distanza, si evince che numeri perfetti pari siano strettamente collegati a numeri primi del tipo *2m-1*, anche detti *primi di Mersenne*. La scoperta di Euler per verificare che il numero appena scritto sia primo si basa sul piccolo teorema di Fermat, che sintetizzato dice che: se *q* è un numero primo, allora *2q-1* è uguale a *1* nel gruppo ciclico con *q* elementi, che significa che sarà pari a *1* ogni *q volte*, ogni qual volta cioè è uguale a *1* il risultato del modulo rispetto a *q*. Quindi Euler dimostrò un teorema enunciato da Fermat. Si inaugurò quella che è conosciuta come *Teoria delle congruenze* che ancora oggi è un campo di grande interesse per la teoria dei numeri.

Aver richiamato la funzione in un ciclo di centinaia di migliaia di iterazioni, è costato un po' di tempo anche per il mio vecchio catorcio di PC, ma ha mostrato la presenza di diverse coppie di numeri amici. Scartando le coppie con cifre uguali, perché perfetti, come 496 che non sono da considerarsi amici perché non diversi; per inciso è stato evitato di fare un controllo che eliminasse tali coppie per non rallentare l'esecuzione; i risultati ottenuti mostrano l'individuazione di 14 coppie riportate di seguito.

220	*1184*	*2620*	*5020*	*6232*	*10744*	*12285*	*17296*	*63020*	*66928*	*67095*	*69615*	*79750*	*100485*
284	*1210*	*2924*	*5564*	*6368*	*10856*	*14595*	*18416*	*76084*	*66992*	*71145*	*87633*	*88730*	*124155*

È interessante notare alcuni aspetti, ad esempio il fatto che una maggiore presenza di questa tipologia si concentra oltre che per numeri relativamente piccoli (cinque sotto il 10000), nell'intervallo di diecimila dei 60000 (ve ne sono ben 4), mentre, si registra la completa assenza tra i 20000 e i 60000.

CONCLUSIONI

Giocare con i numeri fa sempre bene, fidatevi. La sfida di questo mese è approntare una strategia vincente per il gioco del divisore. È ugualmente interessante e divertente scovare nuovi numeri perfetti o coppie di numeri amici, o comunque cimentarsi nell'analisi di quelli già ottenuti.
Buon divertimento e alla prossima!

Fabio Grimaldi

NOTA

CRITERIO DI EULERO

Il criterio di Eulero è molto utilizzato per la ricerca di numeri primi, ovviamente, con l'ausilio del mostro al silicio. Da tale teoria si è ottenuto il numero perfetto più grande ad oggi conosciuto, tranne essere smentito da nuovi studi, che è calcolabile da un primo di Mersenne già conosciuto nel novecento 26972593-1.

Webaccessibile.org

Web-accessibility o accessibilità dei siti Web. Cosa significa? Cos'è l'accessibilità? Come si costruisce un sito Web accessibile? Quali sono le linee guida da seguire? Quali gli strumenti per verificare il lavoro svolto?

Ci si scandalizza, a ragione, davanti ad un pubblico edificio che pone delle barriere architettoniche nei confronti dei cittadini disabili. Ma avete mai pensato che un sito mal progettato possa lo stesso contenere delle "barriere architettoniche virtuali" nei confronti di chi, afflitto da qualche tipo di handicap, abbia intenzione di usufruire delle informazioni e dei servizi offerti? Non tutti usiamo il computer, e di conseguenza il Web, allo stesso modo. Prendiamo in esame una persona afflitta da cecità totale. Vi aspettate forse che usi il PC servendosi di un monitor 17'' con 32 bit di profondità del colore? Ovviamente no: la sua postazione sarà equipaggiata con dispositivi hardware e software appositamente studiati, che permettono l'uso della macchina anche senza il classico dispositivo di output video. Naturalmente anche il browser di questa postazione non potrà essere un browser tradizionale: spesso e volentieri sarà un navigatore a sintesi vocale, capace di leggere direttamente quello che è scritto nei documenti Web visitati. Non credete, a questo punto, che un sito Web basato interamente sulle immagini o sui filmati Flash, senza testi alternativi, senza materiale che il software possa leggere, non potrà rappresentare un'esperienza di navigazione felice per un non vedente? Senza giungere al caso estremo della cecità totale, prendiamo in considerazione una persona che ci vede poco (e se mi levo gli occhiali rientro pienamente in questa categoria). Una pagina Web farcita di caratteri minuscoli non è l'ideale, in casi come questo. Si potrebbe andare avanti per ore: uno scriteriato accostamento di colori potrebbe rappresentare una seria difficoltà per chi, ad esempio, è daltonico.
Cambiamo categoria e passiamo alle difficoltà motorie: chi non usa il classico mouse come periferica di puntamento potrebbe essere messo in crisi da un sito che non abbia previsto questa eventualità. Le possibili barriere architettoniche virtuali sono davvero molte.

LA LEGGE IN MATERIA

Più il tempo passa più i servizi offerti attraverso il canale del Web aumentano. I privati sono stati i primi a muoversi. Pensiamo ai servizi bancari: già da tempo è possibile consultare il proprio estratto conto, fare bonifici, pagare le bollette, ricaricare la carta di credito e quant'altro semplicemente stando seduti davanti ad un PC connesso in Internet. Poi è stato il turno della pubblica amministrazione: finalmente è possibile evitare alcune delle lunghe code agli sportelli servendosi dei nuovi canali messi a disposizione dallo Stato. Non è necessario essere dei profeti per pronosticare che questa tendenza è, o almeno dovrebbe essere, ben lontana dall'arrestarsi. Già, ma come l'ufficio dell'anagrafe non deve essere in cima ad una scalinata sprovvista dell'apposita rampa per chi vi si reca in carrozzella, l'analogo servizio veicolato tramite il Web non deve contenere alcuna fra le barriere architettoniche virtuali citate nel paragrafo precedente. Anche se con un po' di ritardo, pure in Italia siamo arrivati a capirlo, e già da qualche tempo un'apposita legge tutela i cittadini disabili, spingendo per l'accessibilità dei siti Web. Nel caso della pubblica amministrazione sono previste delle sanzioni nel caso di illecito, mentre i privati sono invogliati ad una maggiore accessibilità attraverso degli incentivi.

Fig. 1: ***Risorse in lingua italiana sull'accessibilità dei siti Web***

IL PIACERE DI SVILUPPARE SITI WEB ACCESSIBILI

Indipendentemente da quello che dice la legge, sviluppare siti Web accessibili è un piacere, almeno per chi riconosce in cuor suo il problema. Resta chiaro che un privato è libero di realizzare il proprio sito Web come più lo desidera, anche senza rispettare alcun criterio di usabilità e di accessibilità. Se si vogliono fare le cose per bene, però, non è davvero possibile tralasciare questi aspetti. Le motivazioni sceglietele voi: possono essere tanto altruistiche quanto egoistiche, oppure un mix delle due cose. Non riesco a trovare un motivo valido per non realizzare siti accessibili. Vendete qualcosa sul Web? Un sito inaccessibile significa meno clienti. Non vendete nulla, mantenete un sito solo per il piacere di farlo? Mi sembra evidente che desiderate che le vostre pubblicazioni possano raggiungere quante più persone possibili, ed un sito Web accessibile significa un'utenza potenzialmente più vasta. Avete sviluppato un sito Web solo per voi stessi, perché vi piace rimirarlo? Sarà lo stesso un sito migliore se è anche accessibile.
Se conoscete il problema dell'accessibilità, dovreste automaticamente concludere che sviluppare siti Web accessibili è meglio tanto per chi il sito lo visita quanto per chi lo mantiene.

WEBACCESSIBILE.ORG

Il sito che vi propongo questo mese è una risorsa in lingua italiana sull'accessibilità dei siti Web. Vi trovate tutto quello che concerne l'argomento: le direttive tecniche, le linee guida, le normative di legge, i software validatori ed altro ancora.
Collegata al sito c'è anche una mailing list per discutere il tema. Naturalmente si tratta di un sito leggero, comodo da consultare, usabile ed accessibile.

Carlo Pelliccia

Biblioteca

ON LINE

CODE TOAD

Codice sorgente free, tutorial approfonditi e una miriade di risorse per assistere lo sviluppatore

www.codetoad.com

SUPERDOTNET

Una serie inesauribile di risorse per programmare alla grande con la nuova piattaforma Microsoft .NET (ASP.NET, VB.NET, C#,)

www.superdotnet.com

THE SERVER SIDE

Così come evidenzia per bene il sito: la tua Enterprise Java Community

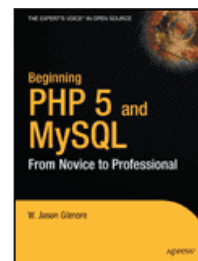
www.theserverside.com

LE SMART CARD, I SISTEMI ELETTRONICI DI PAGAMENTO E LA RETE

Il volume è curato dal "Forum per la Tecnologia dell'Informazione" all'interno di un Osservatorio, oggi arrivato alla quinta edizione, che dal 1989, propone un costante monitoraggio ed aggiornamento degli scenari socio-economico, socio-culturali e tecnico applicativi dei sistemi, dei servizi e degli strumenti di pagamento. Le carte elettroniche, prima, e le smart card, poi, hanno cambiato le modalità di utilizzo del denaro da parte delle persone, per la loro intrinseca capacità d'innalzare le performance transazionali e di sicurezza. In questo testo sono presenti dati particolarmente utili per gli operatori bancari e finanziari, ma anche per le istituzioni, le imprese e per l'opinione pubblica in genere, per articolare l'offerta di nuovi servizi e nuove tecnologie da parte del sistema bancario italiano.

Difficoltà: Media • Autori: Giorgio Pacifici (a cura di) • Editore: Franco Angeli www.francoangeli.it **• ISBN: 88-464-5870-2 • Anno di pubblicazione: 2004 • Lingua: Italiano Pagine: 220 • Prezzo: € 22.00**

BEGINNING PHP 5 AND MYSQL, FROM NOVICE TO PROFESSIONAL

Oltre alle caratteristiche fondamentali del linguaggio PHP, sono analizzate, in maniera specifica, tutte le potenzialità della versione 5, in particolare il migliorato supporto object-oriented, il supporto di SQLite e di SimpleXML. Diversi capitoli sono dedicati ad un dettagliato approfondimento dei concetti basilari della programmazione PHP, altri trattano invece, di argomenti più avanzati come il session-handling, la LDAP integration, i Web Service e le tecniche di programmazione sicura. Oltre al PHP, un ampio sguardo è rivolto anche al potente e performante MySQL database server. Troverete tutto sull'installazione, il processo di configurazione, i datatypes, le chiavi di sicurezza, il data import/export e tutte le principali funzionalità che MySQL può offrire insieme al PHP.

Difficoltà: Media • Autore: W. Jason Gilmore • Editore: Apress www.apress.com
ISBN: 1-893115-51-8 • Anno di pubblicazione: 2004 Lingua: Inglese • Pagine: 736 Prezzo: $ 39.99

SISTEMI OPERATIVI

È il testo ideale per chi vuole apprendere le nozioni basilari e introduttive sui Sistemi Operativi, indicato particolarmente per gli studenti di ingegneria o di informatica. Vengono esaminati concetti comuni a sistemi operativi diversi, illustrandoli tramite casi di studio (Unix, GNU/Linux e Windows) ed esempi.

Il testo fa riferimento a una tradizionale architettura monoelaboratore e a nessun linguaggio di programmazione in particolare, utilizzando uno pseudo linguaggio "C-like". Sono presenti anche due appendici sulle problematiche di sincronizzazione in sistemi distribuiti e sul linguaggio Java.

Tra i principali argomenti trattati: Gestione e sincronizzazione dei processi, Gestione della memoria e delle periferiche, File System, Unix, GNU/Linux e Windows.

Difficoltà: Bassa • Autori: Ancilotti, Boari, Ciampolini, Lipari • Editore: Mc Graw Hill http://mcgraw-hill.it **• ISBN: 88-386-6069-7 • Anno di pubblicazione: 2004 • Lingua: Italiano Pagine: 368 • Prezzo: € 27.50**

INBox
L'esperto risponde...

Java e MySQL

Ciao a tutti, ho un piccolo problema: è possibile da un programma realizzato in Java stabilire una connessione ad un DB MySQL? Se la risposta è si vorrei sapere in che modo procedere.

Christian

La piattaforma di sviluppo Java dispone di una serie di driver (*JDBC*) che consentono di effettuare connessioni a numerosi tipi di database, tra i quali è presente anche MySQL. Segue un semplice esempio di connessione:

```
// ----- MYSQL
String connectionURL =
"jdbc:mysql://localhost:3306/miodatabase?
  user=miousername;password=miapassword";
Connection connection = null;
Statement statement = null;
ResultSet rs = null;
Class.forName("org.gjt.mm.mysql.Driver").
                    newInstance();
connection = DriverManager.getConnection(
              connectionURL, "", "");
statement = connection.createStatement();
```

Infine, non bisogna dimenticare di chiudere tutto nel modo seguente:

```
rs.close();
statement.close();
connection.close();
```

Font di sistema

Salve a tutti e complimenti per il lavoro che svolgete. Sto sviluppando un'applicazione in Visual Basic e avrei bisogno di recuperare la lista dei Font presenti nel sistema, ma non so come procedere. Sapreste indicarmi i passi da compiere? Grazie anticipatamente.

Eugenio

Svolgere questo tipo di operazioni è molto semplice, sono necessarie poche righe di codice, infatti, basta utilizzare la proprietà *Fonts* dell'oggetto *Screen*. L'esempio seguente inserisce la lista dei caratteri in un *ListBox* di nome *List1*:

```
Dim I As Integer
    For I = 0 To Screen.FontCount - 1
    List1.AddItem Screen.Fonts(I)
Next I
```

Acquisizione di stringhe con scanf()

Ho un problema da risolvere, spero possiate aiutarmi. Il "mistero" in questione riguarda l'acquisizione di una stringa, contenente anche spazi vuoti, attraverso *scanf*. Ho provato ad usare sia la funzione *gets(variabile)* sia la funzione *scanf("%[^\n]", &variabile)* ma al momento dell'esecuzione, quando è il momento dell'acquisizione non riesco a scrivere nulla.

Giuseppe

La funzione *gets()* è deprecata, il motivo di tutto questo è che non dispone di controlli per stabilire la lunghezza dell'input immesso. Al suo posto è possibile utilizzare *fgets* col prototipo

```
char *  fgets (char * string , int num , FILE
                              * stream);
```

come mostrato nell'esempio seguente:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_LEN  256
int main()
{  char *str;
   str = (char*) malloc((MAX_LEN+1) *
                        sizeof(char));
   printf("prompt: ");
   fgets(str, MAX_LEN, stdin);
   printf("str: %s\n\n", str);
   free(str);
return 0;
}
```

C e le password

Salve a tutti. Ho appena creato un programma in C per DOS che permette di proteggere alcuni file tramite l'utilizzo di una password. Il problema che non riesco a risolvere riguarda il momento della digitazione della password. Infatti, vorrei che al momento dell'inserimento dei caratteri venga visualizzato il carattere asterisco. Grazie in anticipo.

Andrea

Prova ad usare il codice seguente dovrebbe fare proprio al caso tuo:

```
#include <stdio.h>
#include <conio.h>
#include <mem.h>
main() {
  char password[10];
  char a;
  int i;
  memset(password,'\0',sizeof(password));
  i = 0;
  do { a = getch();
   printf("*");
   password[i] = a;
   i++;
  } while(a != 13);
  printf("\nLa password inserita è:
                        %s",password);
  getch();
}
```

Stampare una form

Complimenti a tutti e grazie anticipatamente per l'attenzione concessami. Vengo subito al dunque. Sto realizzando una semplice applicazione alla quale vorrei aggiungere una funzione che mi permetta di catturare la form in esecuzione e salvare il contenuto in un file *bitmap*. Come devo procedere?

Antonio

Il codice riportato di seguito cattura la form in esecuzione e copia il contenuto nella clipborad in formato *bitmap:*

```
procedureTForm1.Button1Click(Sender: TObject);
var
  imgWindow: TBitmap;
begin
  imgWindow := GetFormImage;
  try
    Clipboard.Assign(imgWindow);
  finally
    imgWindow.Free;
  end;
end;
procedure Tform1.Button2Click(Sender: TObject);
var
  imgWindow: TBitmap;
begin
  imgWindow := TBitmap.Create;
  try
    imgWindow := Form1.GetFormImage;
    imgWindow.SaveToFile('C:\immgine.bmp');
  finally
    imgWindow.Free;
  end;
end;
```

I namespace in C#

Da poco tempo ho iniziato a programmare e leggo con regolarità la vostra rivista, attraverso la quale sto cercando di imparare (almeno lo spero) i segreti della programmazione. Il mio linguaggio preferito è C# e a questo proposito vorrei sapere cosa sono e a cosa servono i *namespace*.

Attilio

I namespace vengono utilizzati per organizzare le applicazioni realizzate in C# secondo diversi criteri di visibilità. L'idea che sta alla base del concetto di namespace non è nuova nel mondo della programmazione, e riprende quella utilizzata dagli sviluppatori del linguaggio Java quando hanno creato i *package*. L'esempio seguente servirà a chiarire meglio il concetto.

```
namespace utility.datastructure;
public class Stack {
   public void push(int value){
      ....
      ....}
   public int pop() {
   ....}}
```

La classe *Stack* è inserita in un namespace denominato *utility.datastructure*. All'esterno del namespace è possibile accedere alla classe *Stack* in due diversi modi. Il primo consiste nello specificare tutto il namespace:

```
utility.datastructure.Stack stack =
                new utility.datastructure.Stack();
```

Con il secondo metodo si dovrà indicare all'inizio del file che si è intenzionati ad usare le classi presenti nel namespace *utility.datastructure*. Questo potrà essere fatto usando la parola chiave *using:*

```
using utility.datastructure;
```

successivamente un oggetto di *Stack* potrà essere istanziato senza utilizzare il namespace:

```
Stack stack = new Stack();
```

L'utilizzo dei namespace non è obbligatorio, ma fortemente consigliato, in particolare nello sviluppo di componenti che dovranno essere successivamente riutilizzati.

Codifica di una stringa

Salve a tutti e complimenti per la rivista. Ho sviluppato una piccola applicazione in Visual Basic e avrei bisogno di inserire al suo interno una funzione che mi consenta di codificare e successivamente decodificare una stringa.
Avete qualche consiglio da darmi?

Mimmo

Il problema può essere tranquillamente risolto utilizzando il codice seguente:

```
Dim Code As String, DataString As String,
                              Temp As String
Sub Translate()
Dim I As Integer
Dim location As Integer
Temp$ = ""
For I% = 1 To Len(DataString$)
location% = (I% Mod Len(Code$)) + 1
Temp$ = Temp$ + Chr$(Asc(Mid$(
                DataString$, I%, 1)) Xor _
            Asc(Mid$(Code$, location%, 1)))
Next I%
End Sub
Private Sub Command1_Click()
'cifratura della stringa, sostituisce
    'abcdefghijk' con qualsiasi altra stringa
                        Code = "abcdefghijk"
DataString = "Il tuo messaggio segreto è "
Translate
MsgBox (Temp$)
End Sub
Private Sub Command2_Click()
DataString = Temp$
Translate
MsgBox (Temp$)
End Sub
```

Verificare la presenza di una tabella in un database

Sto sviluppando un' applicazione Visual Basic che consente di accedere ad un database.
È possibile stabilire se una data tabella è presente all'interno del database?
Se si, come bisogna procedere?

Marco

Ecco un sistema abbastanza veloce per verificare l'esistenza di una tabella all'interno di un database:

```
Private Sub Command1_Click()
   MsgBox
TabellaPresente("tblNomeTabella")
End Sub
Private Function TabellaPresente(tableName
As String) As Boolean
    Dim strConn As String
    Dim rst As Recordset
    strConn =
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & _
            "c:\esempio.mdb"
   Set rst = New Recordset
   On Error Resume Next
   rst.Open tableName, strConn, _
      adOpenForwardOnly, adLockReadOnly,
adCmdTable
   TabellaPresente= Not CBool(Err.Number)
   On Error GoTo 0
End Function
```

PER CONTATTARCI

e-mail: ioprogrammo@edmaster.it

Posta: Edizioni Master,

Via Cesare Correnti, 1 - 20123 Milano